User Manual for PS-I, Version 4.0.4

Dr. Ian S. Lustick

Software Development by

Dr. Vladimir Dergachev

Scripting Manual as Appendix 3 by

Dr. Dan Miodownik

Updated December 2005

Table of Contents

MANUAL FOR USING THE PS-I T00L-KIT, VERSION 4.0.4

Foreword

PS-I (originally standing for "Political Science—Identity) is a sophisticated but easy to use tool-kit for the production of powerful agent-based or computational simulation models. This manual is provided to assist analysts and researchers who wish to learn how to use PS-I to design and produce analytically useful dynamic models of competitive environments. The uses to which PS-I can be put are limited only by the imagination of the user. Models so far produced focus on a wide variety of problems involving political, cultural, psychological, administrative, geographical, and other factors. No programming experience is necessary to use PS-I.

Getting Started

If you would simply like to plunge into using PS-I without producing a customized model you can use one of the models available as defaults. Just follow the directions on p. 1 to download, install and open the program. Double click on model4.mdl, press "View"-→ "New field viewer", and then press "run." A dynamic display will begin.

This manual is offered as both a guide and a reference. It begins with instructions for downloading and installing the executable program from the web and for opening the program itself. From that point the manual is organized as a tour, moving from "button" to "button" to discuss menus and displays presented by the program and how the options they reveal can be used to build simple or complicated "landscapes." Landscapes is an agent-based modeling term of art for the grids upon which present the dynamic results of moving the model forward through time.

Each paragraph is numbered separately. When a topic or term is discussed, relevant paragraphs elsewhere may be included in parentheses to guide the user toward amplified discussion of the topic or term. The italicized term "*Note:*", indicates a helpful hint for using PS-I or or an explanation that stands apart from the flow of the topic being discussed, but which may be important to avoid confusion or misunderstanding.

Appendix 1 contains a glossary of terms used in the design of models with PS-I. Appendix 2 presents a library of commonly used commands for selecting different kinds of "agents" and for making changes to them. These commands draw on a kind of "pseudo computer code" language. It is not necessary to master the "grammar" of this language to use PS-I, but sophisticated users will find themselves naturally extending and applying the regularities apparent in these commands.

PS-I has been under development for more than five years. It is has reached a level of flexibility, precision, and ease of use close to what we have been hoping for. Still we expect updates and new versions to be developed. The current version, available on the web, is PS-I 4.0.4 When new versions are available they will be posted (as explained in the manual). Meanwhile the feedback received from those who use this manual will be of extraordinary importance to me and to others working on PS-I. We therefore encourage your criticisms and suggestions, both of the toolkit and of the manual. We are eager to make corrections and improvements.

Finally I would like to thank and recognize those who have contributed in crucial ways to the development of PS-I. First and foremost I must mention Dr. Vladimir Dergachev. He is a software engineer whose imagination, enthusiasm, and devotion to the highest standards of reliability and precision have made this project possible. Our partnership has been one of the highlights of my career as a researcher and social scientists. Dr. Dan Miodownik has been an outstanding research assistant and collaborator who has contributed immensely not only to the substantive work done with PS-I (and its forerunner, "ABIR"), but also to techniques of script-writing and data manipulation without which systematic work with PS-I for research purposes would not be possible. Benjamin Eidelson has also made extensive contributions to the exploration of PS-I's flexibility, ease of use, and to the design of exciting new features. Dr. Maurits van der Veen has also been a collaborator and source of excellent advice as we have moved through the various stages of developing the tool-kit. Thanks for their special contributions is also due to Dr. Britt Cartrite, Dr. Roy J. Eidelson, and to Kaija Schilde.

Please feel free to contact me with questions, suggestions, or problems that arise. I am most easily reached by email at: ilustick@sas.upenn.edu

Dr. Ian S. Lustick
University of Pennsylvania

MANUAL FOR USING THE PS-I TOOL-KIT, VERSION 4.0.4

A. DOWNLOADING PS-I

1. PS-I is an easy to use tool-kit for the production of a wide variety of agent-based models. The tool-kit is specifically organized to enable its use for the production of virtual arenas of political interaction and contestation. These may be used to study mechanisms of theoretical interest in abstract but well controlled virtual settings, or to map relationships among variables in stylized "historical" or "geographical" settings at different levels of granularity. Landscapes can even be designed to map existing knowledge about drivers and relationships in particular places and times, thereby yielding a virtual "country" or "situation" for experimental purposes.

2. PS-I is easily and freely downloadable from SourceForge at http://ps-i.sourceforge.net/ Users can download PS-I by clicking the "Latest Stable Version." After the installation file is run the program is opened by double-clicking on the PS-I.exe file in the folder created that contains PS-I. PS-I is not currently available for MAC users except via emulation of a Windows, NT, or Linux environment.

3. The earlier version of this software, released as "ABIR" (the Agent-Based Identity Repertoire model) in its various versions, can be understood as a relatively simple, limited capacity model rather easily producible from the PS-I platform. Users not wishing to employ the great range of agent design, micro-rule, and process flexibility available in PS-I may wish to employ the final version of ABIR (ABIR-28). That model is downloadable by clicking the "software" button at http://www.psych.upenn.edu/sacsec/abir/. A manual for its use is also downloadable, including a separate manual for script-writing.

    - *Note*: There is a link at this site to SourceForge where PS-I can be downloaded. This site also has a repository of published and unpublished research done with PS-I as well as templates and raw data needed to replicate our studies.

B. LOADING FILES INTO PS-I

1. After the installation file is run an icon for the PS-I.exe file will be visible within the subfolder containing PS-I. PS-I itself is opened by double-clicking on that file or a shortcut easily created to it. The dialogue box will then appear with "PS-i" in the upper left hand corner. A smaller dialogue box, called "Open," should automatically open as well with a list of six folders. The first folder, labeled "sample files," provides a ready-made set of models (mdl files). Loading any one of these permits the user to create others or to save a landscape as a "snapshot" (snp file [see below]). Alternatively, the user can either create a model with the "New Model Wizard" (see below C1a) or load an existing mdl or snp file into the program. Clicking on either a model file or a snapshot file will load that file into the program. **For the purpose of this manual, we will assume the user chooses and loads "model4.mdl".**

- *Note*: The five other folders contain software files and are not normally accessed by the user.

- *Note*: Loading a snapshot file will load the model (the set of rules for the behavior of agents) which produced the exact configuration as it appears in the snapshot, and a landscape with that configuration in time the snapshot was saved in accordance with the random seeds listed under "model parameters" (see below N).

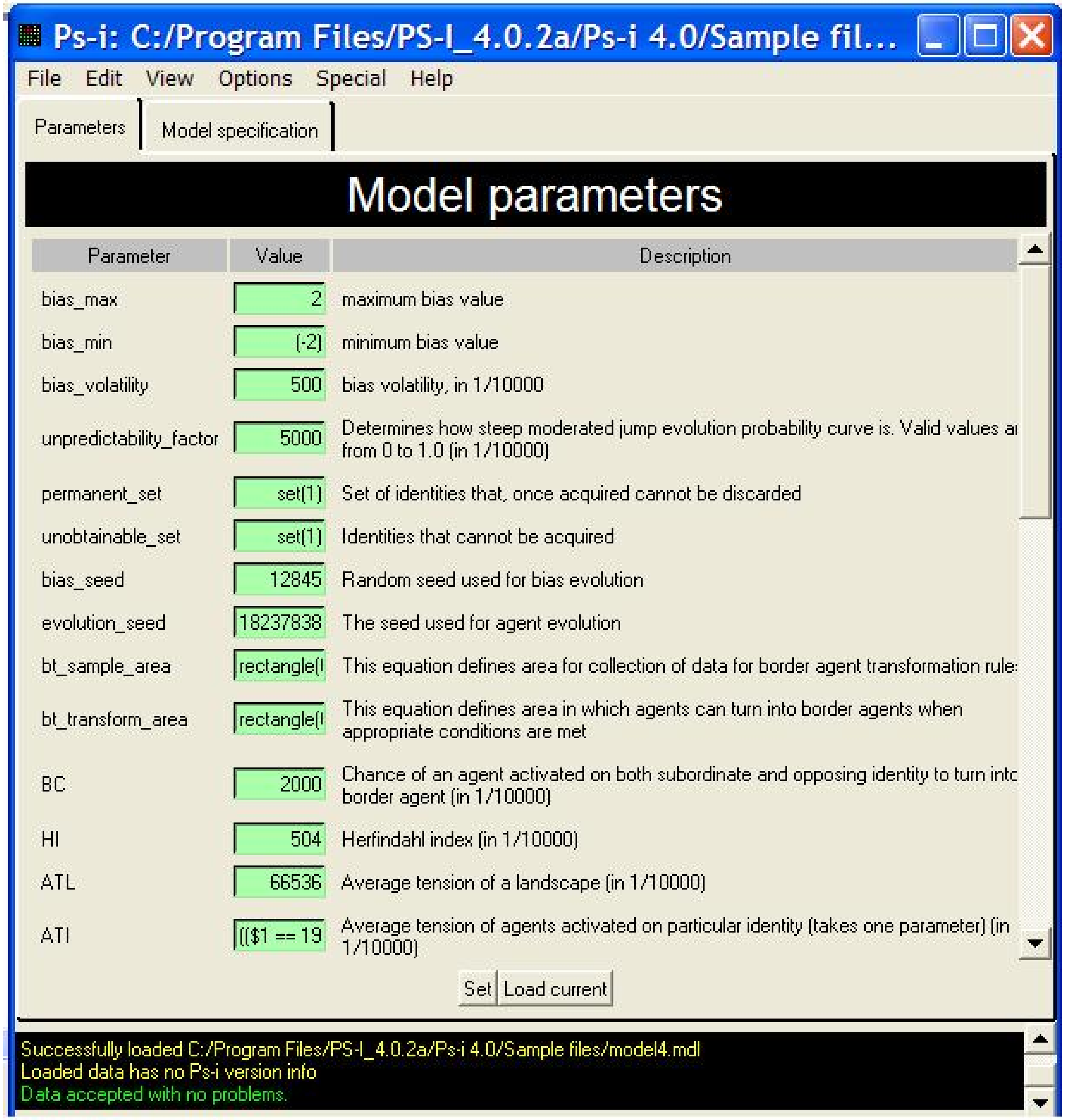
- *Note*: Landscapes saved with ABIR program are "lnd" files. These files can also be loaded into PS-I, which will translate them into snapshots (snp files).

2. For purposes of using this manual the user should highlight model4.mdl, then click the "open" button. You will be prompted to decide whether to allow a program operating with Tcl scripts to do so on your computer. Unless you have some reason to think that the program could have been contaminated (an event that has yet to occur in our experience), then you should click "Yes, don't ask again." This will allow you to use the program repeatedly over a session without having to answer this question. Whenever you load PS-I itself, however, as opposed to loading a file for PS-I to work with, you will be given this prompt.

3. The display that follows your response to the Tcl script prompt is the main screen for viewing, changing, manipulating, and running agent-based models using the PS-I toolkit. In the blue bar at the top, after "PS-I" the pathname of the file loaded into PS-I is displayed. (See Figure 1.)

4. The buttons under the top blue bar in the main PS-I display read: ***File***, ***Edit***, ***View***, ***Options***, ***Special***, and ***Help***. Clicking these buttons opens specialized drop-down menus.

Figure 1: Main PS-I Display



C. MAIN PS-I DISPLAY: FILE BUTTON

"File Button"

1. The "File" button offers the following options:

   a. *New Model*: Create a new model from scratch using the "New Model Design Wizard" to establish or change landscape size, numbers of identities, agent classes, and various parameter settings using a point and click format. This is a quick and easy way to produce a working model, but one operating within a very limited set of parameters.

   b. *Load*: Call up a pre-existing model or snapshot.

   c. *Save Edit Window*: Save the parameter and design settings of a model without saving the particular "geographical" features, i.e. the particular assignments of particular identities as activated or subscribed by particular agents in particular locations.

   d. *Save state*: Save the parameter and design settings of a model as well as its "geographical" features, i.e. the particular assignments of particular identities as activated or subscribed by particular agents in particular locations at whatever point in time (time step) is currently displayed. (time step is displayed in the black box in the center of the bottom bar where it reads:

      "time:" e.g. "time: 0" or "time: 215"

   e. *Save statistics to:* Designate a path name and file name for the export of statistics gathered by PS-I as the landscape moves ahead through time.

   f. *Stop statistics output:* Stops the saving of statistical information to the designated file.

   g. *Exit:* Closes PS-I

D. MAIN PS-I DISPLAY: EDIT BUTTON

Edit Button

1. The "Edit" button offers the following options:

   a. *Find:* Search within the model specification file (see below O)

b. *Field:* Capture, copy, and move selected portions of a landscape, using point and click menus. The options presented here are mostly quite straight-forward. The user can easily insert or delete columns or rows of agents while designing or redesigning a landscape with the Field Viewer (see below F). The most powerful functions here are "Subsampled Resize" and "Subsampled copy." These are convenient techniques for customizing landscapes and are of particular relevance to sections below dealing with use of the Selection Editor (I) and the Effect Tool (K).

c. *Subsampled Resize:* By entering the proper coordinates the user can increase the size of a selected local array of agents ("source block") so that the pattern it contains is reproduced in a larger segment of the landscape ("target block"), which could be in fact the landscape as a whole. Changing the listed "seed" insures randomization of identities activated and contained within each agent's repertoire in the resized area. If the seed is changed, only the particular proportions of agent-classes are reproduced in the resized block, not their location or attribute complexion.

d. *Subsampled Copy:* By entering the proper coordinates the user can reproduce a designated portion or "source bloc" of the landscape somewhere else in the landscape—the "target bloc." This operation overwrites the agents in the target bloc with the content and arrangement of the agents in the source block. If the seed is changed, only the particular proportions of agent-classes are reproduced in the target bloc. If the seed is left unchanged, the target bloc will be an exact replica of the source bloc.

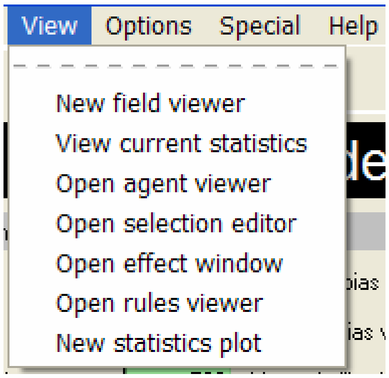
E. MAIN PS-I DISPLAY: VIEW BUTTON

<u>View Button</u>

- *Note*: clicking on the dashed line at the top of the menu underneath the View Button automatically places this menu in the upper left hand corner of the computer display, thereby allowing quick and easy access to the important editing capabilities available within this menu.

1. The “View” button displays the following menu:

Figure 2: View Menu



2. Clicking on these options gives the User access to a wide array of tools for examining and editing PS-I landscapes.

F. FIELD VIEWER

1. *“New Field Viewer”* presents a visual display of the landscape using colors and icons to express the activated identity and agent class of each agent in the landscape.

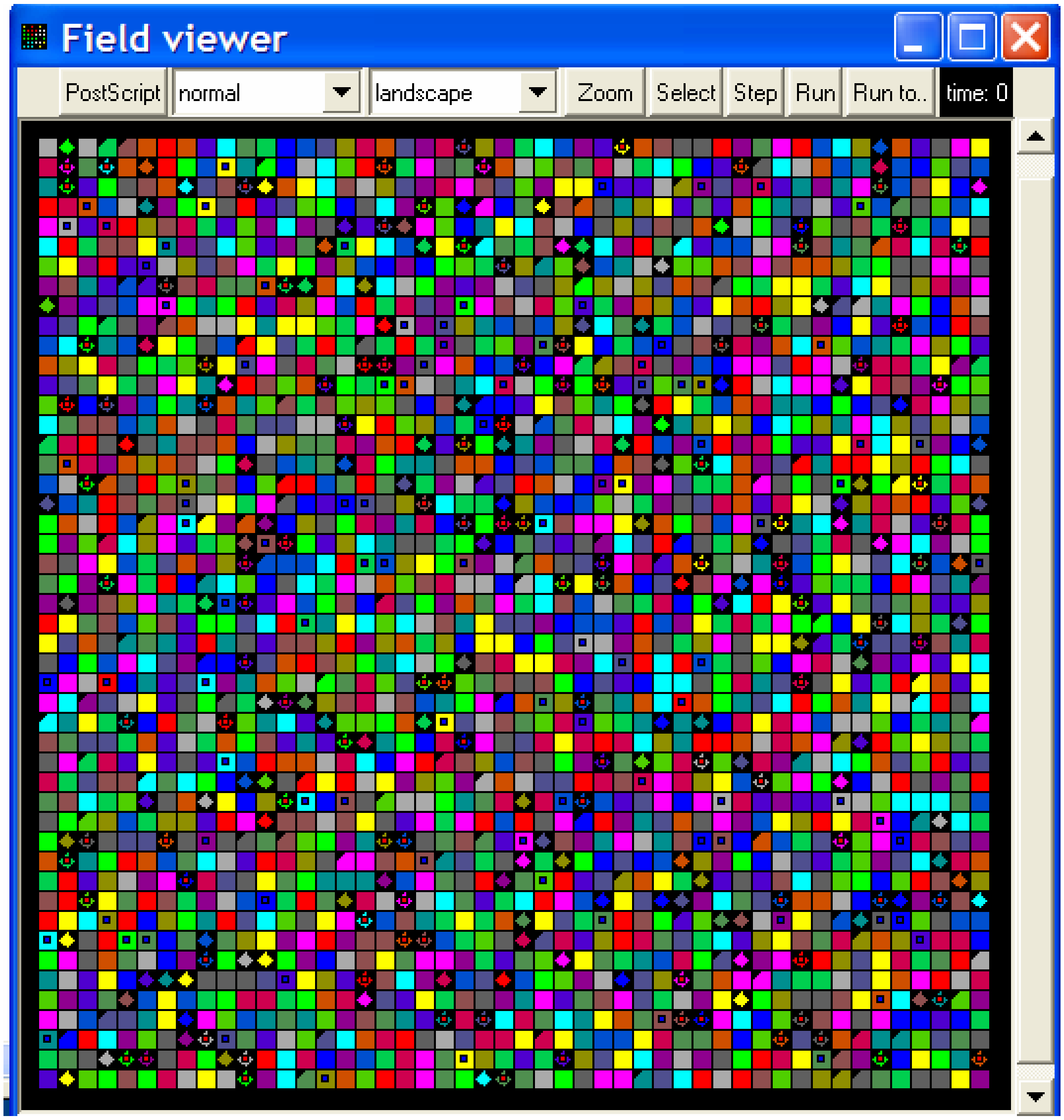
- *Note*: In the models that PS-I displays, and in this manual, the potential attributes of agents contained in their “repertoires” (repertoire is another term for “cache,” as is "subscription") are described as “identities.” This image is drawn from constructivist identity theory—prominent in various forms in psychology, political science, anthropology, and sociology. However, research can and has been done with PS-I that imagines those attributes as any set of traits or versions which can be possessed by individuals or “replicators” (in the context of evolutionary theory) at any level of analysis (states, villages, families, individuals, organisms, germs, etc.). PS-I imagines these units as actively, that is publicly, expressing only one of these attributes, identities, or versions at any particular time. For example, instead of individuals with a repertoire of identities, PS-I can be used to model arguments capable of being understood or made by citizens in a deliberative democracy, or preferences for different sorts of products or candidates held by consumers or voters, or as firms with various corporate strategies at their disposal. The rules governing the activation, abandonment, substitution, or rotation of these traits would then be adjusted to

correspond with the theoretical or consensual knowledge about the domain to be modeled.

2. On a gray bar along the top of the Field Viewer (see Figure 3) there are eight buttons:

- PostScript
- Toggle arrow: normal, tension
- Toggle Arrow: landscape
- Zoom
- Select
- Step
- Run
- Runto

Figure 3: Field Viewer



3. We may leave aside the “PostScript” and “Toggle Arrow: Landscape” buttons.

   a. *Toggle Arrow: normal, tension:* The standard display offered in the field viewer matches colors with the specific numbered identity currently activated by each agent. Switching to “tension” view offers a display which uses different colors to register, for each agent, the amount of “tension” it is experiencing. Tension is the number of agents in the neighborhood of particular agent currently activated on an identity different from the identity activated by that agent. (The size of the neighborhood {range, or “sight radius”} for each kind of agent is stipulated as a parameter setting in any particular model. See below H2 and I3.)

b. *Zoom*: Increases or decreases the size of agents in the display and allows for details to be viewed more clearly in smaller portions of the landscape (close, normal) or for the landscape to be viewed more easily as a whole (Far, Eagle).

c. *Select*: Opens the Selection Editor for identifying (highlighting) the presence and location of groups of agents using a point and click menu or a freehand box for various Boolean expressions. The Selection Editor is also accessed directly through the View menu and will be discussed fully below.

d. *Step*: Clicking this button moves the landscape through time one time-step. The effects of agents interacting with one another according to the algorithms (micro-rules) governing their behavior are visible as changing icons and changing patterns of colors displayed by agents over time. Whether agents update on the even or odd time steps, or according to some other schedule devised by the user, depends on the agent classes to which they belong which in turn are set by the user in designing or changing the update rules for specific agent classes. Changes in the display are presented only after even-numbered time steps.

    i. *Note*: Within the parameter specification file, see below section , the expression "(time mod 2)==1" means that the agent class involved updates on even time steps while the expression "(time mod 2)==0) updates on odd time steps), in other words **before** agents updating on even time steps.

e. *Run*: Clicking the "Run" button will move the landscape through time. A "Run" button is also present on the bottom grey bar of the main PS-I display (see Figure 1). Once the "Run" button is clicked, it changes to "Stop." Clicking the "Stop" button brings the evolution of the landscape to a halt and changes the button back to "Run." Movement of the landscape through time may be resumed by again clicking the "Run" button.

f. *Runto*: Clicking on the "Runto" button opens a dialogue box with the prompt: "Step until" Typing a number into this box and then clicking "Go!" will move the landscape through time and stop it at the time step corresponding to the number typed into the Runto box. When the Runto function is used the landscape does not visually change until the dynamic process reaches its stipulated endpoint. The advantage of this function is that PS-I can run faster than it can when each step is visually rendered.

G. STATISTICS DISPLAY

<u>Open Statistics Viewer</u>

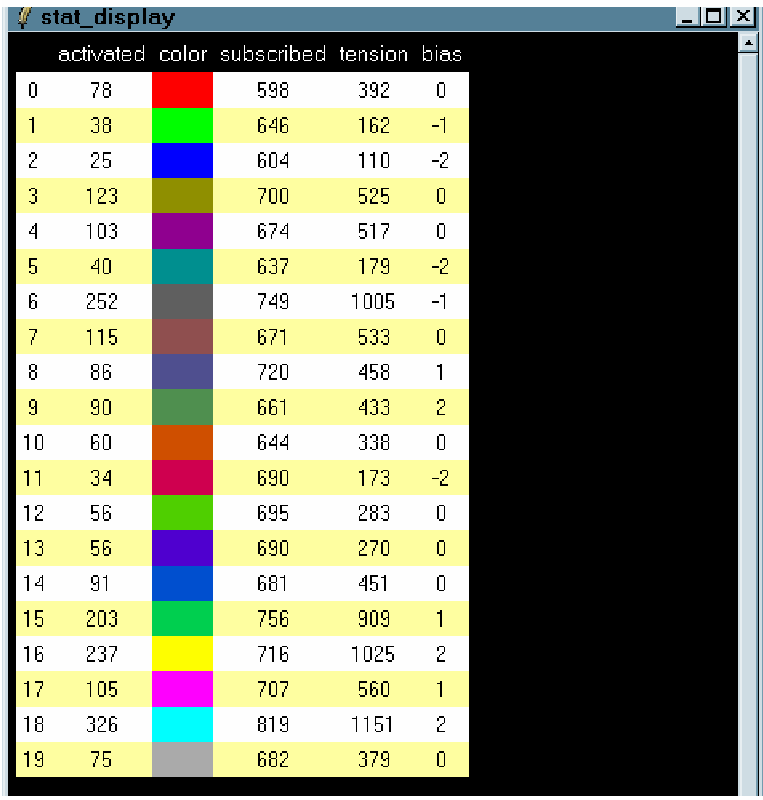
1. Clicking "Open Statistics Viewer" in the "View" menu produces "stat_display." Statistical information about the status of the landscape is gathered by PS-I and the results are registered here. At time "0" for any mdl file the statistics display describes the

result of a randomized distribution of identities and agent types conforming to the rules for the design of landscapes embedded within that particular model (see below M5). This is the information that can be exported in the form of csv files (readable by Excel or SPSS) using the "save statistics" command as explained above (C1d).

2. Stat_display is presented here in three parts. Its vertical length makes it difficult to view it all at once on a computer screen. It is comprised of a grid at the top of the display (Figure 4), a grid beneath it (Figure 5), and a list of statistical "probes" at the bottom (Figure 6).

3. Statistics Display, First (Top) Grid

Figure 4: Statistics Display, First (Top) Grid


stat_display

	activated	color	subscribed	tension	bias
0	78		598	392	0
1	38		646	162	-1
2	25		604	110	-2
3	123		700	525	0
4	103		674	517	0
5	40		637	179	-2
6	252		749	1005	-1
7	115		671	533	0
8	86		720	458	1
9	90		661	433	2
10	60		644	338	0
11	34		690	173	-2
12	56		695	283	0
13	56		690	270	0
14	91		681	451	0
15	203		756	909	1
16	237		716	1025	2
17	105		707	560	1
18	326		819	1151	2
19	75		682	379	0

4. The top grid includes the basic information about the landscape at any particular point in time organized by identity. The numbers beginning with zero and running down a column at the extreme left are the identities available for the population of agents in landscapes produced by this model: Model4. In Model4 there are 20 such identities, and so the numbers in the left hand column run from 0 to 19. The headings across the top of the first part of this display are labels for the columns underneath them. The column labeled "activated" reports, for each identity, the number of agents in the landscape at the

current time step expressing, i.e. "activated on," that identity. The column labeled "color" reports the color displayed by each agent activated on the identity corresponding to the number on the left.

- *Note*: Click on the main PS-I display. Click on the "Options" button and then click "legacy colors." This action reassigns the colors to numbers in a fashion that corresponds to the use of these numbers and colors in most of the simulation work we have done with PS-I and, previously, with ABIR. To aid in communication across research projects we standardly implement "legacy colors" in this way for most uses of PS-I.

5. The column labeled "subscribed" reports, for each identity, the number of agents in the landscape at the current time holding it within their individual repertoires of identities. Whichever identity is currently activated by an agent is included within its "subscription."

6. The column labeled "tension" reports, for all the agents activated on each identity, the sum of the encounters at the current time step in which an agent activated on the identity is in contact with an agent activated on a different identity.

7. The column labeled "bias" reports an integer describing an adjustment in the identity weight calculations of agents with regard to each identity. These numbers change over time at rates and within a range set by the user. They are produced for the landscape exogenously by a random number generator, which is included in calculations by agents. See *Note* below for a detailed explanation of the rules governing agent identity activation change via the computation of identity weights, a computation that includes integrating information about current bias assignments.

    - *Note:* Agents update, that is change or maintain their activated identity based on calculations about the attractiveness of different available identities. For each agent each available identity must be assigned an identity weight. Consistent with the rules governing the behavior of particular types of agents, each agent compares this identity weights and either maintains its activated identity or changes it. Thus, in each time step each agent counts the number of agents in its neighborhood activated on identities to which it may have access. This is the primary element in the production of a local identity weight for each relevant identity, for each agent. Before deciding whether to maintain its current activated identity, or change,??b?? each agent adds the integer in this "bias" column (whether negative, positive, or 0) to the result of its survey of the presence of identities in its neighborhood. Thus a "1" bias for an identity means that each agent counting the number of agents in its neighborhood activated on that identity does so, and then adds "1" to that number—treating the bias as if it were another "basic agent" activated on that identity and present in its neighborhood. A "-1" bias would result in a subtraction of one from each agent's assessment of the "identity weight" associated with that identity, at that time step, in its

neighborhood. Bias thus stands for non-local, generalized knowledge and is available to all “active” agents. See below re active vs. inactive agents (G18).

8. Second Grid in Statistics Display

Figure 5: Statistics Display, Second Grid

	bt_activated	color	bt_tension	bt_opposition_count	bias	ATI	SI	O
0	78		392	56	0	50256	0	0
1	38		162	27	-1	42631	0	0
2	25		110	19	-2	44000	0	0
3	123		525	103	0	42682	0	1
4	103		517	70	0	50194	0	0
5	40		179	26	-2	44750	0	0
6	252		1005	183	-1	39880	1	0
7	115		533	78	0	46347	0	0
8	86		458	73	1	53255	0	1
9	90		433	75	2	48111	0	1
10	60		338	45	0	56333	0	0
11	34		173	27	-2	50882	0	0
12	56		283	41	0	50350	0	0
13	56		270	40	0	48214	0	0
14	91		451	70	0	49560	0	0
15	203		909	131	1	44778	0	0
16	237		1025	169	2	43474	1	0
17	105		560	84	1	53333	0	1
18	326		1151	0	2	35306	0	0
19	75		379	57	0	50533	0	0

9. The second grid is located directly below the first. It displays the information PS-I needs in order to calculate conditions under which agents may transform into “Border agents.” This capability can be used, as we have, to study secessionism since it allows the user to experiment with different circumstances that may be hypothesized to give rise to more or less secessionism. The columns of numbers and colors labeling the different identities are the same as in the top grid in this display.

10. The column labeled “bt_activated” in Figure 5 indicates the number of agents activated on each identity located within the area of the landscape within which the “basic agent to border agent” transformation rule is to be applied. We can see that in Model4, as it is loaded in this example, the rule is set to apply to all agents in the landscape. Hence the numbers listed under “bt_activated” in the second grid are identical to the numbers listed for activation in the top grid.

11. Similarly, the column labeled “bt_tension” reports numbers that are identical to those listed under “tension” per identity in the top grid.

12. The column labeled “bt_opposition_count” indicates the number of agents, activated on the corresponding identity, who do not have within their repertoire the currently “dominant” identity. The “dominant identity” is the identity activated by more agents, at any particular time step, than by any other identity. So at the time registered by this display (t=63) we see that identity “18” with 326 agents activated on it is the “dominant identity.” Hence a zero is listed in this column for identity 18 since, by virtue of their activation on 18, each of these agents has the dominant identity in its repertoire.

13. The column labeled “bias” repeats the information available in the top grid.

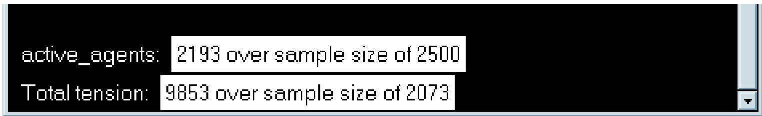
14. The column labeled “ATI” (Average Tension Identity) shows a number that describes the average tension per agent activated on each identity in units of 1/10000. Thus “50256” for identity 0 indicates an average tension level for agents currently activated on identity 0 of 5.0256. (This information is not currently used in the rules that transform agents into border agents.)

15. The column labeled “SI” registers either a “1” or a “0” for each identity. A “1” indicates that for that identity and in that time period the identity is considered a “subordinate identity.” That means 1) no fewer than ten per cent of the agents in the landscape are activated on that identity; and 2) that the identity is not the dominant identity, i.e. it is not currently the identity activated by a plurality of agents in the landscape. To be eligible to be transformed into a border agent the agent must be activated on an “SI” identity.

16. The column labeled “OI” registers either a “1” or a “0” for each identity. A “1” indicates, for that identity and in that time period that the identity is considered an “opposition identity.” That means no more than 20% of the agents activated on that identity have within their repertoires (subscriptions) the identity which, at that time step, is the dominant identity in the landscape. To be eligible to be transformed into a border agent the agent must be activated on an identity which is “OI” as well as “SI.”

17. Below the second grid appear statistics that can be gathered at the request of the user. In this configuration of model4, two additional statistics are being tracked and reported here. Depending on queries inserted into the model specification file by the user there may be many more statistics than appear here. See Figure 6.

Figure 6: Statistics Display, Bottom List of Statistics

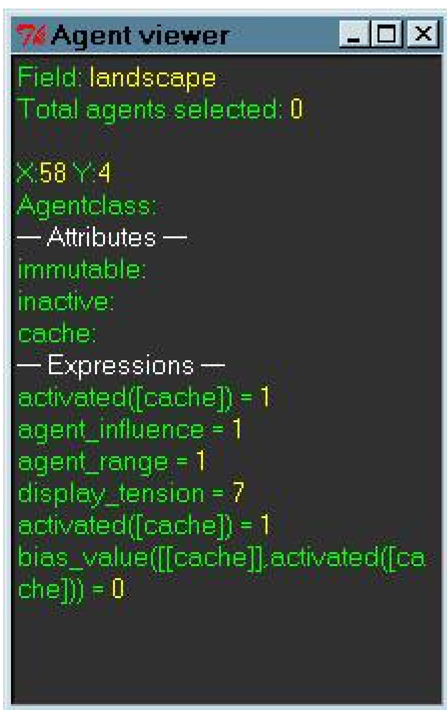


18. The first, "active_agents" indicates the number of agents in the landscape whose activated identity is registered by agents in the local neighborhood. "Inactive" agents are agents that may or may not be affected by their surroundings (depending on whether they are mutable or immutable), but whose activated identity is not taken into account by other agents in its neighborhood.

    - *Note I*: "Border agents" or "Border cells" are inactive as well as "immutable." Colored black, they are neither counted in the identity weight calculations of their neighbors (hence "inactive") nor can their identity activation change in response to outside events. Indeed their activated identity is really only a trace of what identity was activated when they transformed into border agents or were created as border agents at t=0 (time step 0).

H. AGENT VIEWER

Figure 7: Agent Viewer



<u>Open Agent-Viewer</u>

1. Clicking on this button opens a viewer listing information about individually selected agents. See Figure 7. When the viewer is opened it defaults to a view of the agent located at the coordinates X:0,Y:0"—the top left corner of the landscape. By opening the field viewer the user can position the cursor on any individual agent. As the cursor moves from agent to agent the information in the Agent viewer changes to correspond to the particular agent located at the coordinates at which the cursor is pointed. The coordinates are positive on the X axis, but negative on the Y axis, meaning that the agent located at X5 Y23 is to be found by moving six columns to the right (starting from zero and from the top left corner) and then, again starting from 0, 24 rows DOWN.

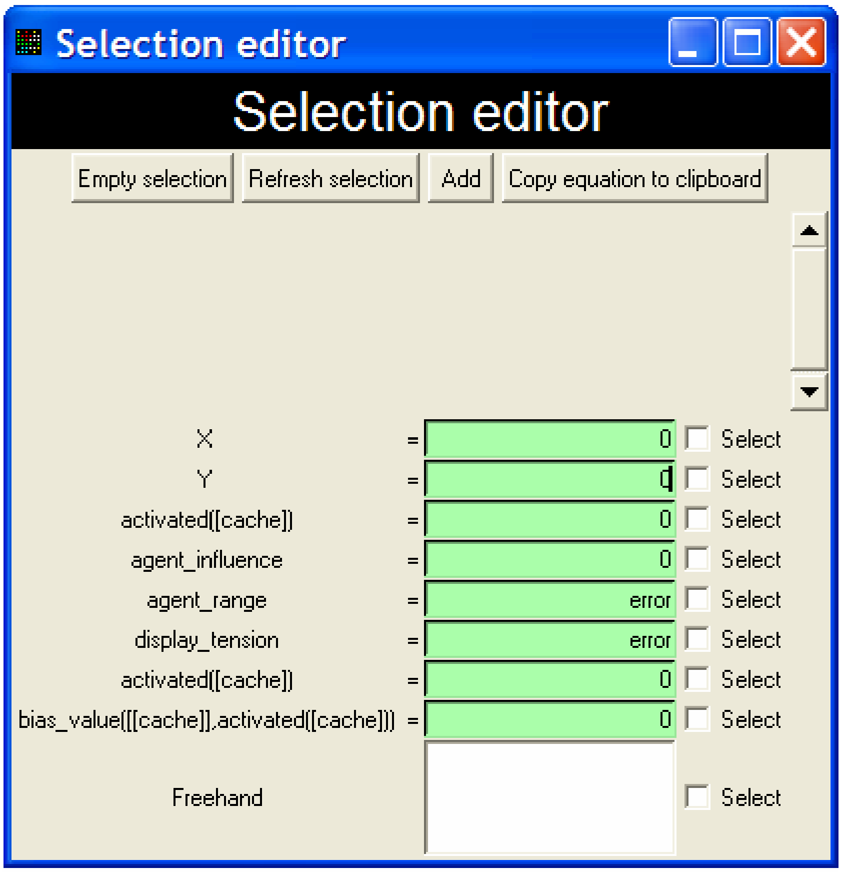
2. Information about the selected agent appears in the Agent viewer beginning at line 3 and includes the agentclass (type) of agent; whether the agent is immutable and/or inactive; what identities are present in the agent's "cache" or subscription. The viewer also reports the activated identity, the agent's influence level (see below regarding influence), its range (the radius of its neighborhood), its current tension level, and the bias currently registered for its activated identity.

    - *Note I*: The second line in the display, "Total agents selected," reports the result of use of the Selection Editor—the next button in the View menu. The number appearing here is the total number of agents in the landscape currently selected.

I. SELECTION EDITOR

<u>Open Selection Editor</u>

Figure 8: Selection Editor



2. 1. The Selection Editor (see Figure 8) is a tool for selecting batches of agents with particular characteristics or locations, either to count them or to change something about them. It is used to "highlight" the batches of agents selected. When an agent is selected and "highlighted" a black and white checkered pattern replaces its activated color, helping it to stand out from the other colored agents of the display. The total number of agents so highlighted appears on the "Total agents selected" line in the Agent Viewer.

3. The Selection Editor presents a vertical array of eight point and click options. By typing an integer into the space provided and then checking the small box to its right, the user selects, or highlights, the group of agents in the landscape meeting the condition thereby stipulated. For example, the first line in the array is "X", indicating the value on the X-axis. Typing "4" (without the quotation marks) in the space provided in this line and then checking the box to its right selects all agents having the value of "4" on the X-axis.

These agents comprise the column of agents 5 in from the left (five and not four because the first column has the value of 0 on the X-axis). These selected agents can be viewed as highlighted in the Field Viewer. Once highlighted in the Field Viewer, the Agent Viewer can be used to learn the exact number of agents highlighted, i.e. the number of agents meeting the stipulated condition.

4. Other point and click options in the Selection Editor include:

   Y = Y-axis

   Activated([cache]) = Activated identity

   Agent_influence = influence level of agent

   Agent_range = radius of local neighborhood "seen" by agent, in other words the radius of the concentric squares surrounding the agents comprehending all the agents that agent is monitoring as within its neighborhood.

   Display_tension = tension experienced by agent (number of direct encounters With local agents activated on a different identity)

   bias_value([[cache]],activated([cache])) = agents activated on an identity with a particular, specified bias

5. The point and click selection options listed above may be combined. When values are stipulated for more than one condition, and each is checked, the resulting group of selected agents is the intersection of these two conditions. Thus if agents activated on identity 6 are selected, as well as agents experiencing a tension level of 5, then only those agents which are BOTH activated on 6 and experiencing a tension level of 5 will be highlighted.

6. At the bottom of the selection editor is a box labeled "Freehand." This space can be used to describe a customized set of conditions not available in the list of point and click options. The user can use simple arithmetic expressions to submit queries regarding attributes of agents in the landscape. When the syntax of the selection command written in the free hand box is correct and understandable to PS-I, the background of the Freehand box turns green, instead of white (default) or red (indicating an incomplete or incorrectly written command).

7. A library of commonly used selection commands is included in Appendix 1. Individual commands can be linked with one another in the Freehand box with "and" or "or" expressions to indicate a conjunctive or disjunctive relationship among the conditions. For example, by typing the following expression in the Freehand box and then clicking the small box to its right, all agents in the landscape containing identity “6” in their repertoires are highlighted.

[cache] intersect set (6)

If the user then types "3" into the activated option in the Selection Editor, and then checks the box to its right, PS-I highlights each agent activated on identity 3 and having identity 6 in its repertoire.

8. The Selection Editor also contains four gray buttons immediately below the black bar with the heading "Selection Editor." The purpose of these buttons is as follows:

   a. ***Empty selection***: click to remove the checks from all the "select" boxes, thereby removing the highlighting of all agents in the landscape and changing the number of selected agents listed in the Agent Viewer to "0."

   b. ***Refresh selection***: click after changing a condition in the Selection Editor to update the batch of agents being highlighted in the landscape and registered as such in the Agent Viewer.

   c. ***Add***: click to write geographical or agent attribute related conditions for selection.

   d. ***Copy equation to clipboard***: click to store syntactical expression of selection in RAM. The expression may be pasted into any word processing program file, or in other parts of the PS-I program (e.g. the effect tool--see below K), using Ctrl-v.

      - *Note I*: The clipboard option allows the (syntactically correct) expressions that appear automatically when the Effect tool is used (see below K2) to be collected, stored, or employed immediately as pasted freehand expressions in the Selection Editor.

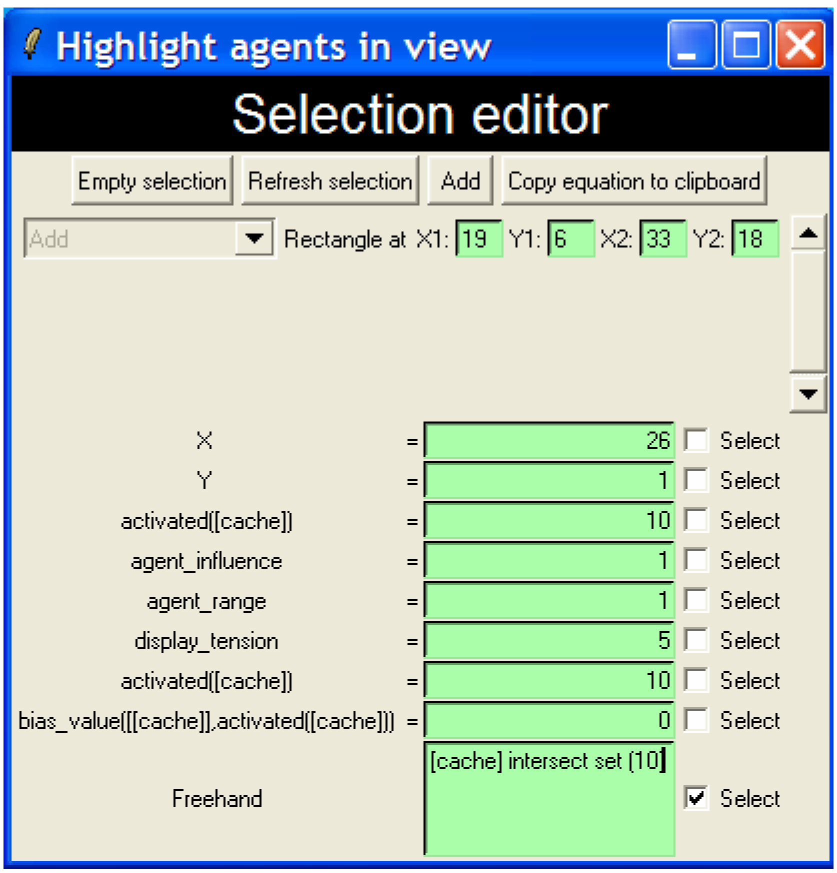
J. SELECTION OF AGENTS WITH CURSOR

Using the Cursor to Select and Highlight Regions of a Landscape

1. A particularly convenient way to select regions of a landscape is to use the cursor on the landscape itself. The user opens the field viewer and points the cursor at an agent to be selected. Right-clicking on that agent produces a list of options for selecting or editing (refashioning attributes of) the selected agent. Within this list are options permitting the user to drag the cursor to produce a rectangle of desired size with the originally selected agent as its upper left hand corner, or to produce a circle of desired radius with the originally selected agent as its center. The group of agents selected can be winnowed and shaped by by then using “subtract” options to remove smaller circles or rectangles from the larger portion selected.

2. Each selection command implemented in this way is registered in the Selection Editor. The coordinates of regions selected appear in the space below the gray buttons in the Selection Editor. In Figure 9 we see that the cursor has been used in the field viewer to draw and select a rectangle whose upper left hand corner is at 19,6 and which extends horizontally and vertically to the agent located at 33,11. Additional commands entered into the point and click selection menu or into the Freehand box will be combined as "and" commands to intersect with the cursor-demarcated regions as the set of agents "currently selected." Using "or" commands expands the selection to include both of the categories described. Multiple regions may be cursor-demarcated. The drop down arrow to the left of any one of them can be used to subtract or add that region from or to a complex selection command. Thus these selection commands will highlight all the agents and only the agents within the rectangle 19,6 33,11 that have identity 10 in their repertoire.

Figure 9: Selection Editor with Region and Agents Subscribed to Identity 10 Selected

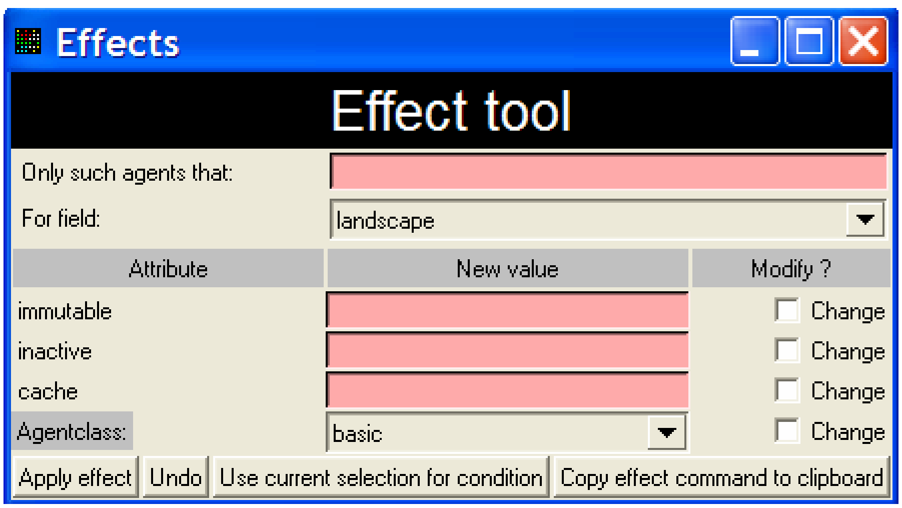


K. EFFECT TOOL

Open Effect Window

1. The Selection Editor plays a crucial role in PS-I, allowing the user to examine which kinds of agents with what attributes exist in what patterns in a landscape at different times. But the Selection Editor plays an equally important role helping the user implement desired shapes, patterns, and distributions of agents in a landscape. This is accomplished with an Effect Tool—a tool for implementing changes in the attributes of any set of designated agents (See Figure 10). An easy way to designate the set of agents to be operated on in this way is by first selecting that set with the Selection Editor and using the Effect Tool to import the designation of that set into the appropriate field in the Effect Window.

Figure 10: Effect Tool

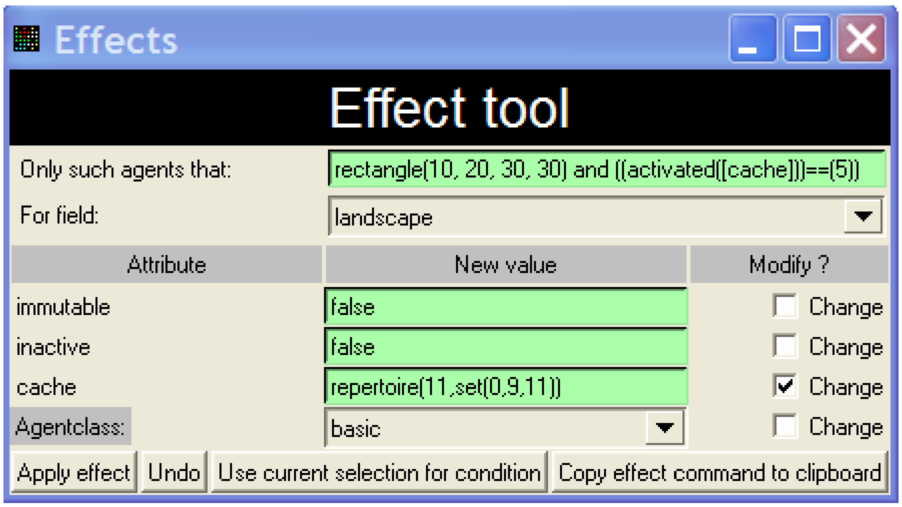


2. In the Effect Tool there is a space labeled: "Only such agents that." Here the user enters a description of the set of agents to be operated on. However, the syntax for properly describing a complex set of attributes or locations can be tricky. That is why it is often more convenient designate agents to be operated on by first highlighting them with the Selection Editor. Once that is done, the "Use current selection for condition" button in the Effect Tool can be pressed. That automatically and accurately registers in the "Only such agents that:" box the description of the set of highlighted agents stipulated in the Selection Editor. The effect command is entered into the Effect tool as some combination of changes in the attributes and/or agent class of the agents selected.

3. For example, in Figure 11 we see that all agents activated on identity 5 and located in region 10,20 to 30,30 are to be subject to an effect command. The effect command to be implemented is that all the agents fitting this description should be changed so as to be activated on identity 11, with identities 0, and 9 also in their repertoires. Note the command

to this effect entered into the “New value” column, in the space to the right of “cache” (in the Attribute column). Note also the checked box on the right of the command labeled “change” in the “Modify?” column. To actually implement the command, to “put it into effect,” the user clicks the “Apply effect” button. For a library of commonly used effect commands see Appendix 1.

Figure 11: Effect Tool with Command to Seed with Agents Activated on 11 with 0 and 9 also in Repertoire



- *Note*: By entering “true” or “false” in the immutable or inactive boxes the user can adjust these attributes of selected agents. Commands applied in this way will not change the behavior of agents whose agent class forbids such a change.

4. In the design of landscapes it is common to want to apply an effect command to only a portion of the agents selected. For example, a user might want to include a particular identity in the repertoires of 50% of the agents within a designated region or in the repertoires of 50% of the agents that already have some other identity in their repertoire. Whatever the particular command to be implemented, it can be implemented on randomly selected proportion of the agents described in the “Only such agents that” line by adding the following phrase:

    and (rand<5000)

5. All proportions in PS-I are expressed as a fraction of 10,000. So adding this phrase will insure that the operation will be implemented on a randomly chosen subset of 50% of the agents included within the “Only such agents that” description.

6. Clicking on the arrow in the "Agentclass" window presents the user with a list of available agent classes. This permits the user to transform selected agents from their agent class to chosen agent class.

L. AGENT CLASS EVOLUTION RULES

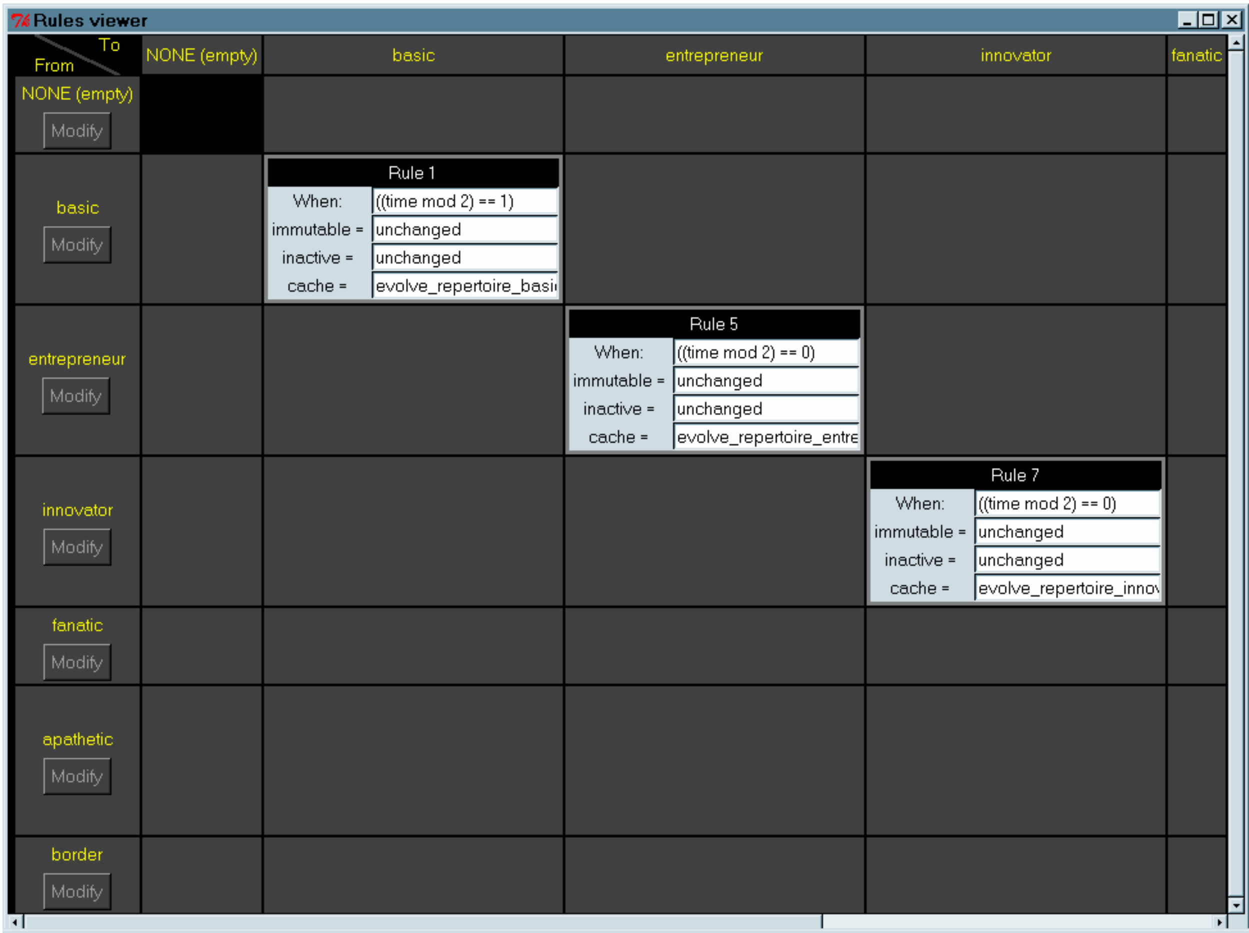
Open Rules Viewer (*Advanced technique, only an introduction*.)

1. Each agent in a PS-I landscape belongs to a particular agent class. The class to which an agent belongs endows it with certain attributes and constrains it or empowers it to act in certain ways in response to various sets of circumstances. Thus a "fanatic" agent (as implemented in model4) may have some similarities to a "basic" agent, but unlike a basic agent it is "immutable" (its activated identity is permanently activated) and the influence of its activated identity is felt by agents in its neighborhood as "3" in identity weight calculations rather than as "1" as is the case with a basic agent. Different icons are assigned to different agent classes so that they appear with an appropriate distinctiveness in the field viewer.

2. The model specification file (viewable by clicking the tab next to the "parameters" tab on the main PS-I display) includes specification of the attributes of different agent classes available in the model that has been loaded by the program (here, model4.mdl). Since political transformations can involve changes in the predominance of different types of agents or increases in certain kinds of behaviors (such as fanaticism), PS-I allows the user to indicate the kinds of circumstances, or rules, that would produce agent transformation or "evolution" such that an agent would be changed from one agent class to another, while retaining the complexion of identities present at that juncture in its repertoire and as its activated identity.

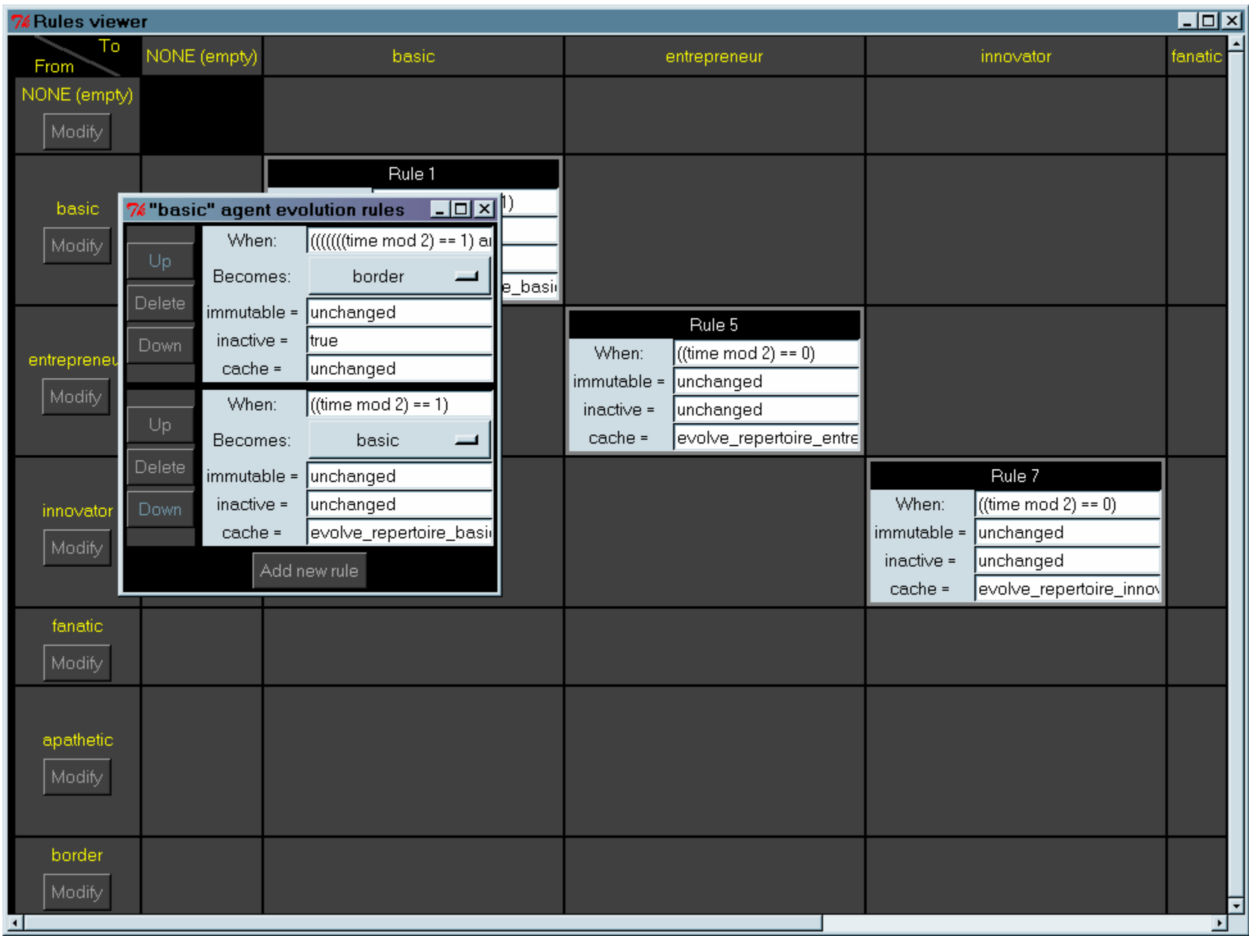
3. The Rules Viewer is a convenient way to review and edit the rules that govern these transformations. Of course it is perfectly possible for a model to have no such rules, in which case an agent that begins a "run" as one agent type will remain that agent type until the end of the run, no matter what happens. Figure 12 shows how the Rules Viewer appears when the Open Rules Viewer button is clicked.

Figure 12: Rules Viewer



2. The grid displayed in the Rules Viewer lists agent classes vertically and horizontally. To read and change rules governing transformation of agents from one class to another the user clicks the "modify" button underneath the name of the agent class in the vertical list which he or she desires to read or edit. Figure 13 shows the result of clicking the modify button associated with the "basic" agent type.

Figure 13: Rules Viewer with Display Produced by Clicking "Modify" Button for Basic Agents



3. We see here that basic agents can turn into border agents under the conditions specified in the box (only partially visible) to the right of the word "When" in the upper half of the display. By editing that text, or the text appearing to the right of the words "immutable," "inactive," and "cache," the rules for the evolution of basic into border agents can be adjusted. Once the new text has been written, closing the box then implements the rule by changing the text as it appears in the model specification file (see below O).

M. NEW STATISTICS PLOT

1. The Statistics Plot window is opened from the "New Statistics Plot" window as displayed in Figure 14. It enables simultaneous graphical display of selected measures of activity in a PS-I landscape moving forward through time. As a default the window opens with the number of agents activated on identity "0" as the measure to be displayed. The "Add curve" and "Remove" buttons are used to remove this measure, replace it with something else, or add additional plot lines. Available choices for statistics to be graphically displayed are presented by clicking the button on the left which reads, as a default, "activated." See Figure 15 for the list of statistics available for plotting in model 4. In this display the user has selected “subscribed,” meaning that the number of agents with the particular identity selected in their repertoire will be tracked and plotted at every time step of the dynamic run.

Figure 14: Statistics Plot

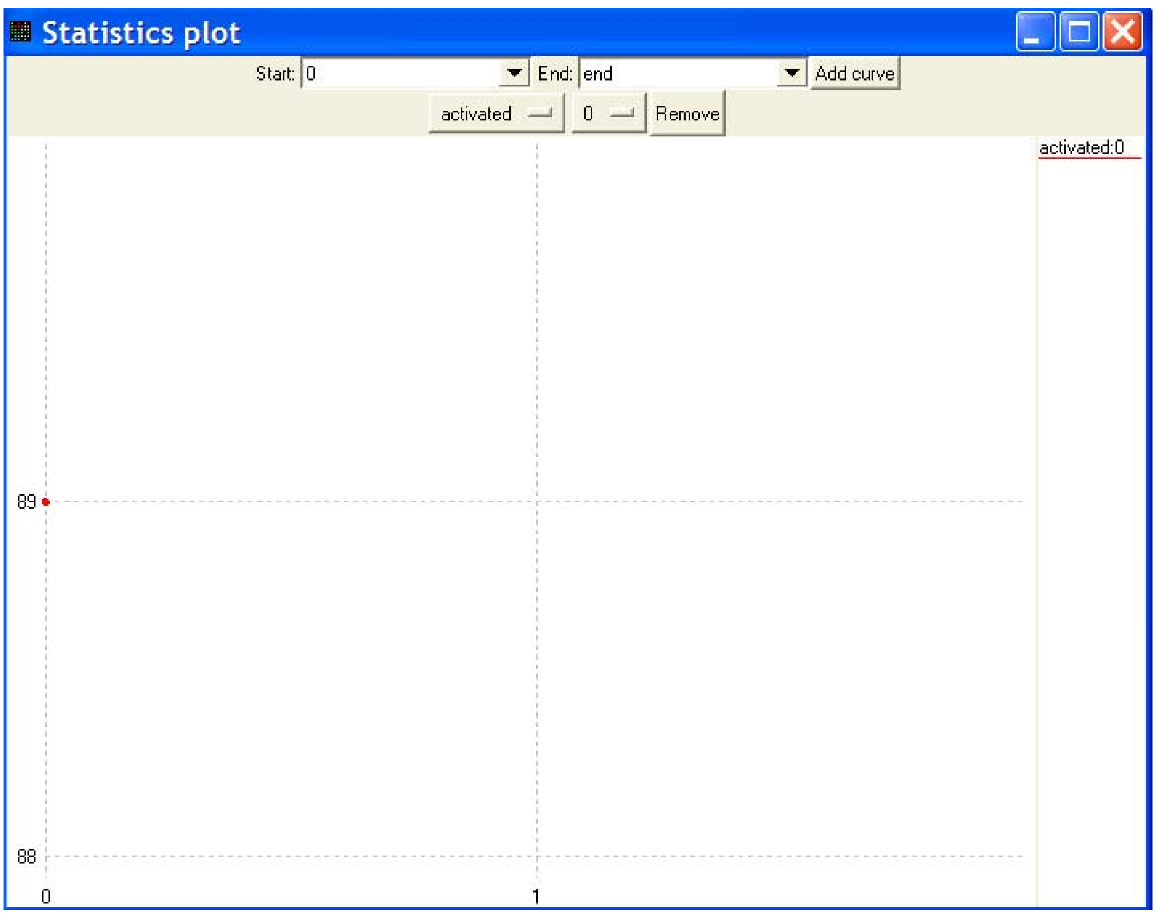
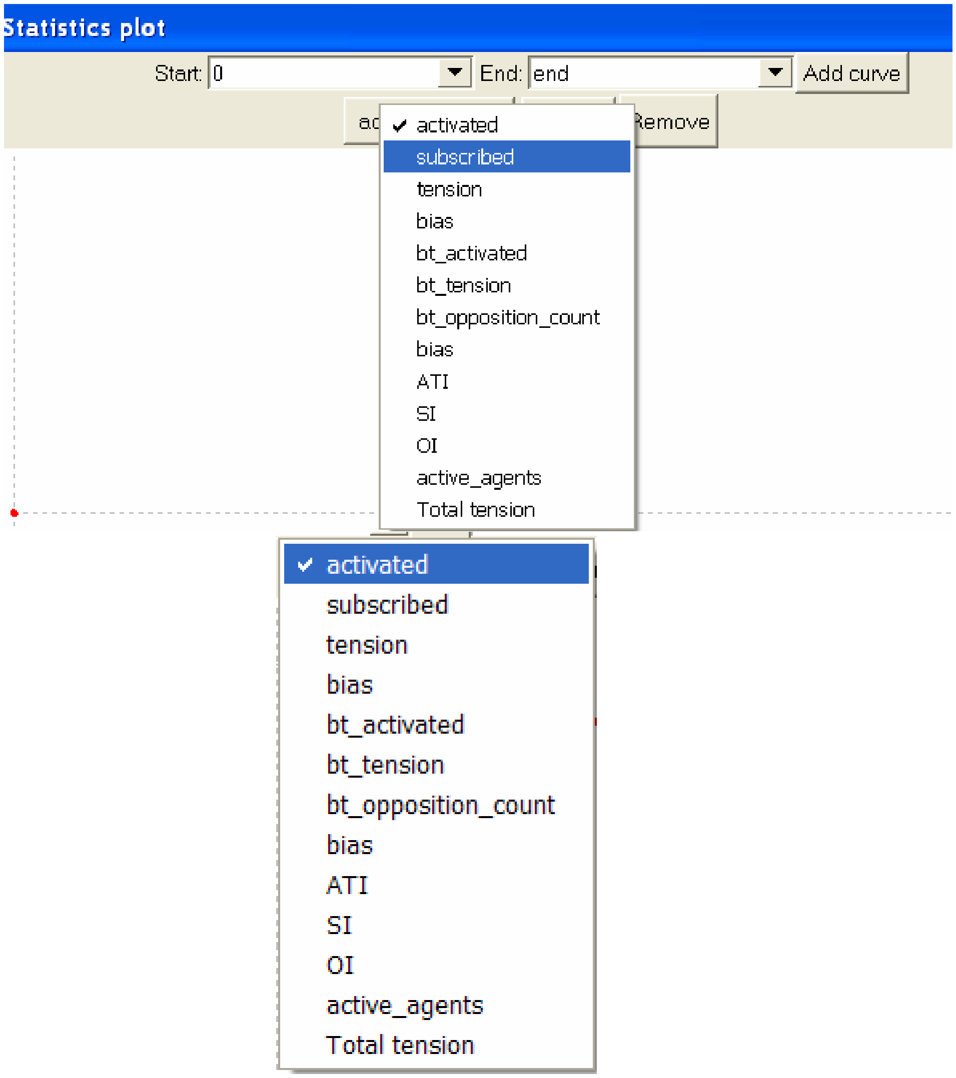


Figure 15: Available Statistics for Plots



Note: The list of available statistics for graphical display is generated by the list of statistics in the model specification file of the model currently in use and can be adjusted as discussed in section O.7 below.

2. The length of the history displayed is determined by the choice made using the "History size" button accessible in the "Options" Window. See below, section N. 1e and Figure 16. The length chosen is the length that will be displayed at any one time. While the run is in progress the user can type a number into the "Start" or "End" window to determine the particular portion of the run to be displayed. If nothing is typed in either of these windows data will be displayed continuously from time 0.

N. MISCELLANEOUS OPTIONS

Options Button

1. The drop down menu that appears when the Options Button is clicked features five options.

   a. *Wrap*: This refers to the length of lines in the built-in text editor. Not normally of importance.

   b. *Acceleration*: The three options provided can be used to adjust the speed at which the program moves through time. In current versions of PS-I the default setting is on "ASM" which provides the most speed.

   c. *Concurrency*: The numbers here refer to the number of processors in the computer running PS-I. The setting is normally "1," but PS-I has can greatly increase its speed by utilizing multiple processors when available.

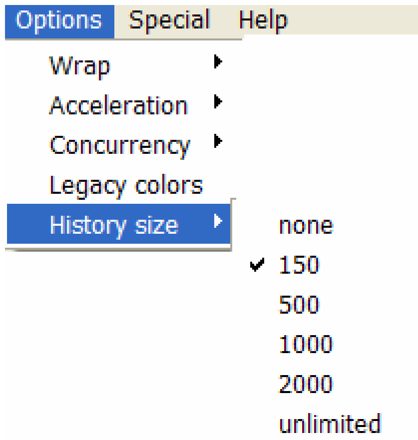
   - *Note*: PS-I performance may be affected when the "ASM" setting is used in combination with settings instructing PS-I to exploit multiple processors.

   d. *Legacy Colors*: When PS-I is loaded particular colors are linked to particular numbered identities. By checking "legacy colors" the colors linked to the numbered identities are changed to correspond to the colors "historically" used by earlier versions of ABIR and PS-I—colors which feature consistently in the research and consulting studies that have been produced with this simulation platform. Normally we do check "legacy colors."

     - *Note*: This is not the case with regard to the colors used to code higher and lower levels of tension in the "tension" display, accessible from the Field Viewer. For the colors "historically" used in ABIR for tension displays, the user must uncheck "Legacy Colors," i.e. use PS-I's default settings.

   e. *History size*: When this button is clicked a drop down list of history sizes appears. See Figure 16. Use this list to decide how big a segment of a model run can be displayed using the "New statistics plot" (under the Options button) as the model runs forward. On most computers PS-I's speed will decrease somewhat as the size of the history it is required to "remember" increases.

Figure 16: History Size for Plotting

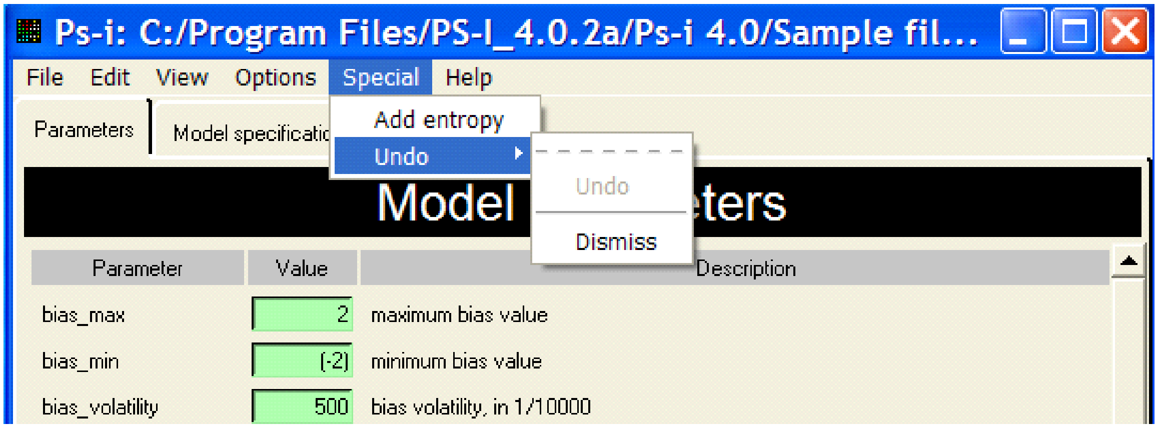


Special Button

1. *Add entropy:* As PS-I landscapes proceed through time they experience changing streams of signals regarding the "biases" (see above G6-G7) exogenously assigned to different identities. These streams of numbers are produced randomly. If "rand" is typed into the Model Parameter boxes labeled "bias seed" and "evolution seed," then clicking this "Add Entropy" button several times insures the virtually complete randomization of the numbers PS-I will use.

2. *Undo:* By clicking "undo" the user can eliminate the consequences of the last "effect command" applied by the user to make adjustments to a landscape. For convenience the "Undo" menu can be detached and dragged to anywhere on the computer screen for easy access during the design of a PS-I landscape. This is accomplished by clicking on the hyphenated line above the "undo" option. See Figure 17.

Figure 17: Undo Menu



<u>Help (Some advanced features)</u>

3. Clicking the "Help" button on the main PS-I display reveals three options: About, Routine Browser, and Color Palette. Clicking "About" displays the version of PS-I being used along with the URLs necessary to download new upgrades and/or the software behind the executable program. Clicking "Routine Browser" displays a list of categories to be used for viewing a glossary of terms recognized by PS-I and includes information about proper syntax in the use of these terms. Clicking "Color Palette" the colors available in PS-I for different identities.

- *Note*: These colors are labeled by the numbers which default to those colors. In other words, this is <u>not</u> a list of the color-number combinations featured in "legacy colors."

4. This completes our discussion of the functions accessed via the row of buttons immediately beneath the top blue bar in the main PS-I display (see Figure 1). We proceed now to the buttons at the bottom the display, to the left of the time display. These buttons are labeled as follows: "Reseed," "Open field view," "Step," "Run," and "Run to…"

<u>Reseed</u>

5. Each model, including model4, has within it a specification of the types and proportions of

agents present in the landscape but does not specify their exact location or identity complexion. "Seeding" a landscape can be understood as substituting agents with particular identity types and repertoire complexions for those already present as a result of the array created automatically when a user calls up a particular model, such as model4.mdl.

6. For example, a user could completely seed a landscape with basic agents activated on identity 5, and with identities 0, 5, and 9 in their repertoire. This action, which could be

accomplished with the Effect tool, would create an absolutely uniform landscape with every agent being identical in every way. To seed a landscape in this way at a rate of 50% would be to transform approximately 50% of the agents in the landscape, chosen randomly, into agents having this particular agent type and identity complexion.

7. Users will often want to compare how landscapes perform over time despite changes in the initial spatial relationships among agents of different types and despite changes in the particular distributions and combinations of identity complexions among agents. This can easily be accomplished with the "Reseed" command. By clicking the "Reseed" button the user generates a landscape in which the proportions of agent types, the size of repertoires of agents, and the number of different identities available to agents, are held constant, but in which the exact distribution and exact number of agents of different types has been randomized along with the particular assignment of identities to the activated and subscribed positions in the repertoires of different agents.

<u>Open Field View</u>

8. Performs the same function as the "New Field Viewer" option in the View menu (see above F1). Clicking it opens a Field Viewer displaying the current configuration of agents.

<u>Step</u>

9. Clicking the "step" button advances the landscape by one time step. It will be observed that PS-I registers change in the activation of agents on every even-numbered time steps even though agents in some agent classes (innovators and entrepreneurs in Model4) may update on odd time steps. Agents updating on even time steps take the updates of those agents activating on odd time steps into account as well as agent updates that occurred on the previous time step.

<u>Run</u>

10. Clicking the "run" button performs the same command as this button on the Field Viewer. It sends the landscape on its journey through time. After being clicked it reads "stop," indicating that if clicked again it will stop the landscape's journey.

<u>Run to…</u>

11. Clicking the "Run to.." button opens a small dialog box. The user types a number into the space provided—a number corresponding to the number of time steps into the future the user wishes the landscape to travel. The user then clicks "go." PS-I will move the landscape through history and then stop automatically at the time-step indicated. (See also, above, F. 3f)

O. MODEL PARAMETERS

1. This completes our consideration of the buttons at the bottom of the main PS-I display. We proceed now to the model itself, accessed underneath the large black banner that reads: “Model Parameters."

2. The display of parameters and their settings is associated with the tab labeled "Parameters." These "model parameters" are windows into the model itself. They allow the user convenient access to most of the important characteristics of the model. (The model itself is viewable (and may be edited) by clicking on the "Model Specification" tab [see below O6-O7].) The vertical scrolling bar on the right edge of the display allows the user to view the descriptions and editing windows of all parameters accessible without actually entering the model specification file itself. Each small editing window features, to its left, the expression in PS-I syntax that describes that parameter. To the right of each editing window is a capsule description of the parameter. For a more detailed explanation of these parameters, i.e. those accessible without editing the Model Specification File directly, see Appendix 2.

   - *Note*: Models may be more or considerably less elaborate than model4. Less complex models will have fewer different parameter settings and fewer edit windows in this display. More complex models will feature parameter settings not present in model4.

3. These parameters can be changed by placing the cursor inside the box next to the parameter one wishes to adjust. With the cursor inside the box, the numbers or text currently appearing can then be either edited or erased and retyped. If this is done correctly the background of the box, which turns red when the editing process begins, will return to green. At this point the change can actually be entered into the model by clicking the button at the bottom of the display labeled "Set." The change will **not** take affect unless the “Set” button is clicked. If the run command is then entered, PS-I will incorporate the changes made via this model editing process into the rules used to drive the landscape through time. If the user wishes to retain the adjusted parameter settings for future use he/she must save the model, using the "save model" command, under another name (otherwise model4.mdl will be overwritten).

4. Load current: Next to the “Set” button at the bottom of the model parameters display is a button that reads “Load Current.” As long as the “Set” button has not been pressed, the user who introduces changes in parameter values into the model parameters windows but who wishes to restore the original values can do so by pressing the “Load Current” button.

P. READING AND EDITING THE MODEL SPECIFICATION FILE

Model Specification File

1. The edit windows listed under “Model Parameters” in the main PS-I display, and explained in

detail in Appendix 2, are, as noted above, actually "windows" looking into the model specification file—the file containing all the (non-hard coded) instructions for PS-I concerning the particular capacities, attributes, and rules comprising the model that has been loaded.

- *Note*: The software code for PS-I itself is available for downloading at SourceForge.

- *Note*: If the field viewer is opened and a particular random array of agents is produced, this landscape may be saved as a "snapshot" from the File menu using the "save state" command. The snapshot (an "snp" file rather than an "mdl" file) thereby created can subsequently be loaded into PS-I with the load command (B1*Note*). The snapshot file contains a model specification file with information identical to that present in the model with which the snapshot was created. It also contains a long list of coordinates and numbers that specify what agents have what identity complexions. The snapshot will thus retain the time step and all the information about the landscape available in the statistics file at the time when the landscape was saved as a snapshot. All the same rules for editing model parameters and model specification files apply whether or not the file loaded was an mdl file or an snp file.

In every model there some parameters are not listed in the array of windows that constitutes the "model parameters" display. These parameters can only be edited by entering the model specification file. By clicking on the "Model Specification" tab in the main PS-I display the user can view and scroll through the entire model specification file. Instead of using the edit windows available in the "model parameters" display it is sometimes easier, for more substantial editing purposes, to find the line in this file with the information reported in that edit window, and edit it right in the model specification file.

2. Specific text within the model specification file can be found clicking the "Edit" button and using the "Find" command.

3. Once the specific text is found and edited, the changes are implemented by clicking the "submit" button at the bottom of the model specification file window. The button will turn red, and if the changes have been properly edited the model parameters will appear with the new value (if it is listed in the parameters) appearing in the appropriate edit window box. Syntactically improper entries will result in an error message. The location within the model specification file where the error was made will be highlighted in blue.

4. Changes properly entered into the model specification file will be incorporated into the rules used by PS-I for any "runs" produced until the program is closed. But unless the changes are saved under the same or a different file name using the "save state" or "save edit window" command, then these changes will not be retained if the model or snapshot is loaded again.

5. Here is an example of editing the model specification file. Suppose the user, interested in secessionist processes, desires to exclude an area surrounding a "state" (the international environment) from the calculations PS-I performs in regard to the production of border agents. This could be done in the model parameter window located along side bt_sample_area. But the edit window is small, and this might be easier if performed inside

the model specification file itself. The user first clicks the model specification tab, then Edit---Find. The user then types the phrase “bt_sample_area” into space provided (without the quotation marks), and clicks “find.” The cursor is brought to the following text:

```
routine 'bt_sample_area' parameter
        comment 'This equation defines area for collection of data for border agent
transformation rules'
        code
"rectangle(0,0,field_width-1, field_height-1)"
end
```

Most of the model specification file is structured in this way. The first line names the rule or routine that is the concern of this particular piece of quasi-code. The “comment” line reports, in regular English, what this particular code is about. The “code” line has the actual specification which appears in the edit window of the model parameters display. The word “end” simply ends this segment of the model instructions. Editing is performed on the code line, which here specifies the coordinates of the rectangle within the landscape monitored by PS-I for the information it needs to determine which identities to designate as qualifying for “subordinate” and “opposition” identity status. As explained above (C4), in Model 4, the entire landscape is monitored, from the origin, to the largest vertical and horizontal values available. That is the meaning of “field_width-1, field_height-1”. To limit PS-I’s monitoring only to a state located within a surrounding belt of “outside-the-state” agents, we can change this line to read: “rectangle(5,5,50,50)” Once the “submit” button is clicked, these new values will be readable with the edit window in the model parameters display.

6. At the very end of the model specification file is a list of statistical probes. These commands instruct PS-I as to the data to be collected and displayed in the csv files importable into Excel or SPSS and produced with the “Save statistics…” command (C1e). The commands included for the collection of statistical information in Model 4 are as follows (copied from the end of the Model4.mdl model specification file).

```
distribution      'activated'      20 "applicable" "activated([cache])"
dist_color                  'color'    'activated'       "$1"
dist_sum                    'subscribed'  'activated'  "([inactive]==false) and is_element($1,[cache])" "1"
dist_sum_single             'tension'         'activated'  "([inactive]==false)" "display_tension"
dist_probe        'bias'    'activated'  'landscape' 0 0  "bias_value([[cache]],$1)"

distribution      'bt_activated'    20 "applicable and bt_sample_area" "activated([cache])"
dist_color                  'color'    'bt_activated'     "$1"
dist_sum_single             'bt_tension'      'bt_activated'  "([inactive]==false) and bt_sample_area"
          "display_tension"
dist_sum_single             'bt_opposition_count'  'bt_activated'  "([inactive]==false) and (is_element(DI,
          [cache])==false) and bt_sample_area" "1"
dist_probe        'bias'    'bt_activated'  'landscape' 0 0  "bias_value([[cache]],$1)"

dist_probe        'ATI'     'bt_activated'  'landscape' 0 0  "ATI($1)"
dist_probe        'SI'      'bt_activated'  'landscape' 0 0  "SI($1)"
dist_probe        'OI'      'bt_activated'  'landscape' 0 0  "OI($1)"
sum               'active_agents' "1" "[inactive]==false"
```

```
sum                 'Total tension' "[inactive]==false" "display_tension"
```

The first command listed here is “distribution ‘activated’”. It results in a list of columns of numbers for every time step indicating the number of agents activated on each of 20 identities.

In general, information collected by using the “distribution” command answers questions such as: “How many agents were activated on identity X at time Y?”

“dist_probe” command produces information about each of the categories collected by the “distribution” command. Examples of data collected by the “dist_probe” command may include the bias for each identity group (what was the bias for identity X at time Y?), ATI, OI, SI (Did identity X qualify as ATI (or OI or SI) at time Y?). “dist_sum_single” presents aggregated information across the identity spectrum. It sums up into the columns values calculated for each agent activated on the specific identity. So to answer a question such as “what was the tension for identity X at time Y?” the statistics monitor collects information for the tension for each agent activated on identity X, sums it up and assigns it to the appropriate cell. “sum” collects global information, for instance “how many agents were active at time Y?”, “what was the total tension at time Y?”

- *Note*: Editing the model specification file can be frustrating and requires some experience. Therefore, it is often helpful to copy the text to be edited, saving it as a record under a different name, and then working on a duplicate of it in Notepad, Wordpad, or Word. Once the editing is completed the text can then be reinserted into the model specification file. This procedure offers the added advantage of preserving a stored a copy of the original text that can be re-inserted if the editing process fails or when the user wants to return to the original setting. Almost anywhere in PS-I it is possible to copy and paste texts, within the program or between PS-I and other programs, using Ctrl-c for copying and Ctrl-v for pasting.

Q. DESIGNING EXPERIMENTAL LANDSCAPES USING “DIAGRAMS” MODE

Introduction

1. Users may wish to use PS-I to produce a wide variety of models for exploring political and other relationships in domestic and international settings. So far PS-I has been used to design templates for experiments of many kinds including studies focusing on:

   - Globalization of different kinds and intensities and its impact in relation to the relative porosity of state boundaries
   - Immigration and the propensity of states to produce populist anti-immigrant reaction
   - Regime repression vs. US diplomacy as techniques for protecting friendly Middle Eastern regimes against internal threats associated with Israeli-Palestinian violence
   - Vulnerability of authoritarian regimes to different kinds of ethnic or religious mobilizations

- Deliberative democracy's contribution to the stability of pluralist orders under varying conditions.
- Prospects for the spread of a "European Identity" under conditions of increasing porosity of separate state boundaries in Europe
- Effects on regime prospects and prospects for terrorist linked nuclear events in Pakistan of Muslim Fundamentalist mobilization in Pakistan under various conditions
- Effects of regime responsiveness vs. repression and autonomy schemes on the likelihood of secessionism and secessionist movements among regionally concentrated ethnopolitical minorities.
- Tipping and cascade patterns of change under different topologies, interaction rules, and network designs.

2. To conduct experiments on these and other similar questions requires production of templates carefully designed to incorporate what is believed to be true about attributes and dynamics of interaction within the political arena or type of political arena being modeled. Thus users may wish to model political arenas characterized by particular distributions of political attachments, geographically concentrated groups, and particular kinds of hierarchical, federative, or other governing structures. Users may wish to focus strictly upon internal dynamics or might also want to be able to locate the arena or arenas they construct within a suitable regional or "global" environment.

12. Governing structures within a polity can be created as networks or webs of influential agents (agents with an influence level greater than 1) who share activated and some subscribed identities and who produce the effects of control and manipulation of a population in the areas over which the network, or web, of such agents is spread. Such "authority structures" can be modeled as dense or lightly present, as hierarchical, pyramidal, or top heavy, as alienated from or responsive to the political identities and commitments of constituents, etc.

13. Examples of such a landscapes are given below in Figures 18 and 19. The webs of icons indicate authority structures attached to government or non-government influentials. Patterns of color indicate concentrations of activation on various salient political attachments.

Figure 18--Beita, Secessionism, t=10

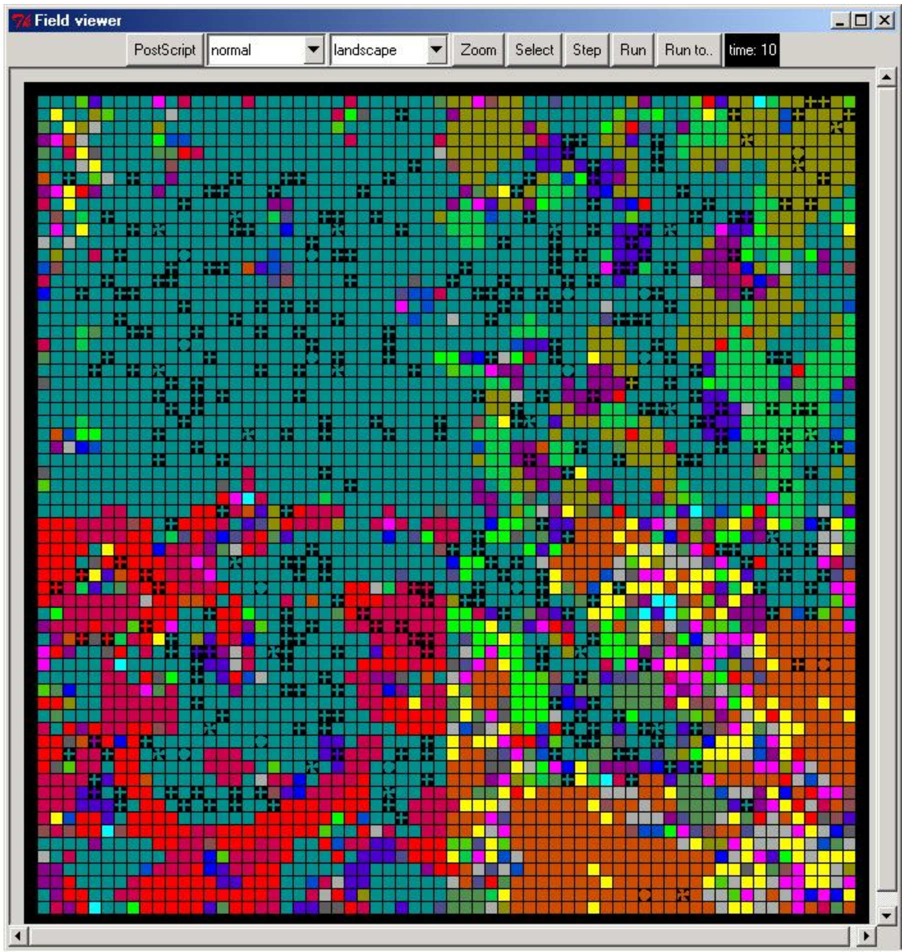
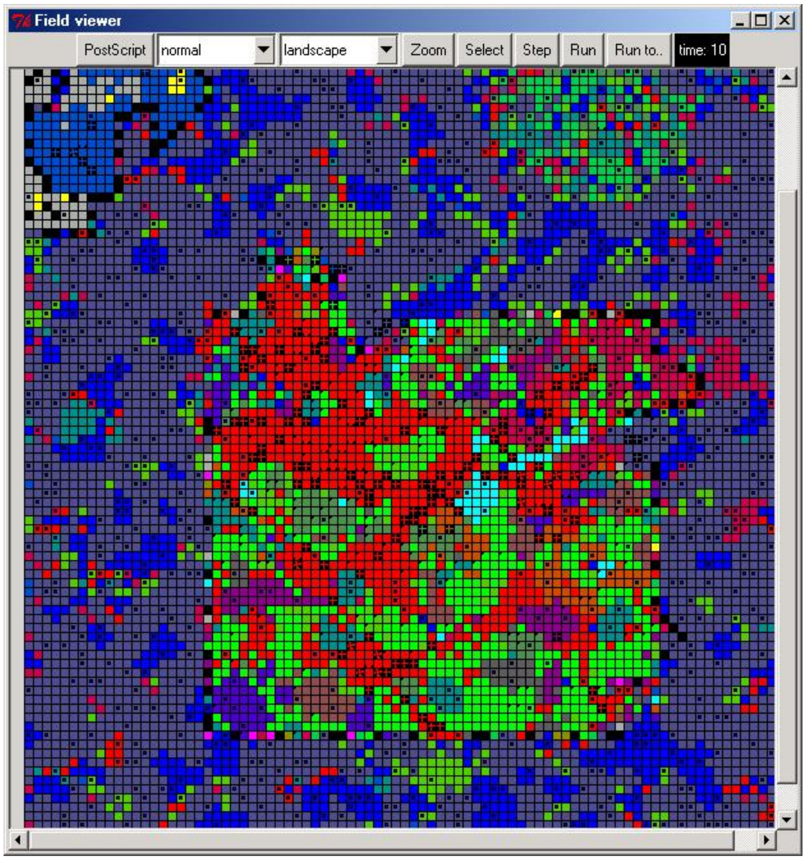
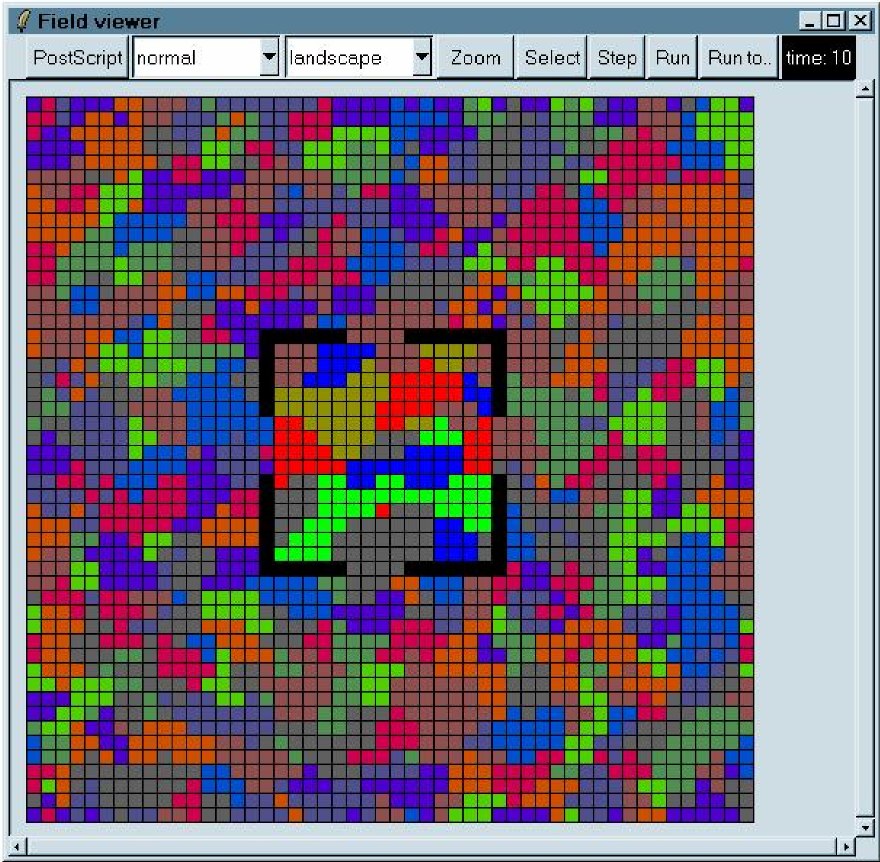


Figure 19—Middle East Polity, t=0



14. Many landscapes useful for experiments are much simpler than this. Figure 20 displays a model of an abstract country with a partially open border subjected to globalizing pressures. It has a “dominant identity” within the boundaries of the polity, but no authority structure, per se, is present.

Figure 20--Globalization, Moderately Porus Boundary, time=10



15. All landscapes can be built from scratch using a basic model to begin with, such as Model4. (Simple landscapes can be produced by using the "New Model Wizard.") Details, including complexions and locations of groups of agents and desired authority structures, can be introduced either with the selection editor and effect tool or by using the field editing capabilities described above (paragraph C1a) or by using the "Diagrams" function for building virtual "polities" with a simple array of point and click menus.

Designing Landscapes in Diagrams Mode

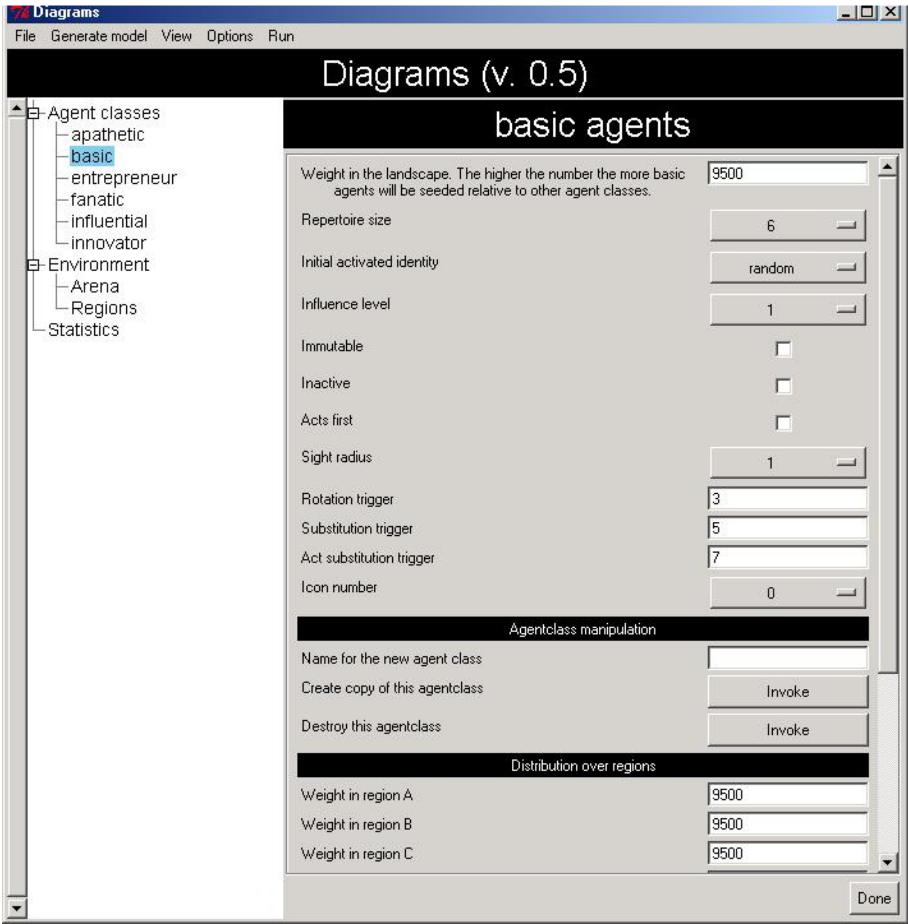
16. The Diagrams display is opened by clicking the Load and then File buttons. From the load file window the user clicks on "Sample Files," then on the "Files of Type" arrow, and then on "Ps-I script[*.scp]". By then double-clicking on "diagrams.scp" the user calls a display entitled "Diagrams." This display offers opportunities to use point and click options to adjust three different kinds of aspects of the model to be produced:

- Agent Classes
- Environment
- Statistics

Agent Classes

17. Under "Agent Classes" the user finds a default list of agent classes available. Clicking on any one of these allows the user to adjust virtually all the elements that determine the behavior and capabilities of agents in that class. For example, by clicking on "basic agent" the following screen appears (Figure 21):

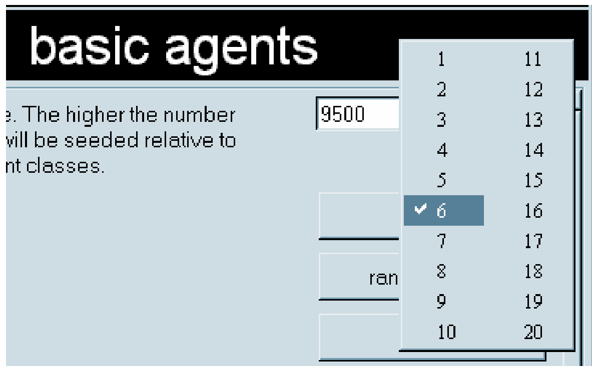
Figure 21: Adjusting Basic Agent Characteristics in Diagrams Mode



18. The 9500 figure listed in the "weight in landscape" box at the top of the list of agent class characteristics is an approximation of the default seeding proportion for the landscape as a whole. It means that if the default model were produced (by clicking the "generate model" button on the gray bar at the top left of the display) approximately 95% of the agents in the landscape would be basic agents.

19. Key aspects of the definition of a basic agent are adjustable with the buttons listed vertically. Thus by clicking the "Repertoire size" button the default repertoire size of "6" for basic agents can be adjusted by clicking on any of the numbers displayed. Figure 22 shows the display that appears when the Repertoire size button is clicked.

Figure 22: Adjusting Repertoire Size of Basic Agents—Diagrams Mode



Each aspect of a basic agent is similarly adjustable, including, as listed:

- Initial activated identity
- Influence level
- Immutable status
- Inactive status
- Acts first (updates on odd time steps)
- Rotation trigger
- Substitution trigger
- Act substitution trigger (directly substituting an identity from outside the repertoire for the activated identity)
- Icon number

20. Under the black bar labeled "sight radius" is located an enhanced tool for establishing the zone of knowledge about other agents that agents in each agent class are to have. The default setting is "no multiple sight zones" (explained soon, below) and a "sight radius" of 1. "Sight radius" is another term for "agent_range." The "range" of a particular agent class (that is its "sight radius") refers to the size of the neighborhood that agents who are members of that class can see. For example, if the range of agents in the "basic" agent class is "1," that means that in addition to seeing itself, each basic agent also sees the activated identity of each of the eight agents immediately touching it, on its sides and corners? If the sight radius is "2," that means that in addition to seeing itself and the eight agents touching it, the central agent also sees the sixteen agents touching the agents that touch it, for a total of twenty-five agents in its neighborhood.

21. Similar options are available to the user by clicking on any of the other default agent classes. However, the user can also create an entirely new agent class. If desired, the settings implemented for any particular agent class, including basic agent, can be used to begin producing a new agent class with a new name, a distinct icon, and a specially design collection of features. This is accomplished as follows. First implement the desired settings. Then type a name for the new agent class. Then click the "Invoke"

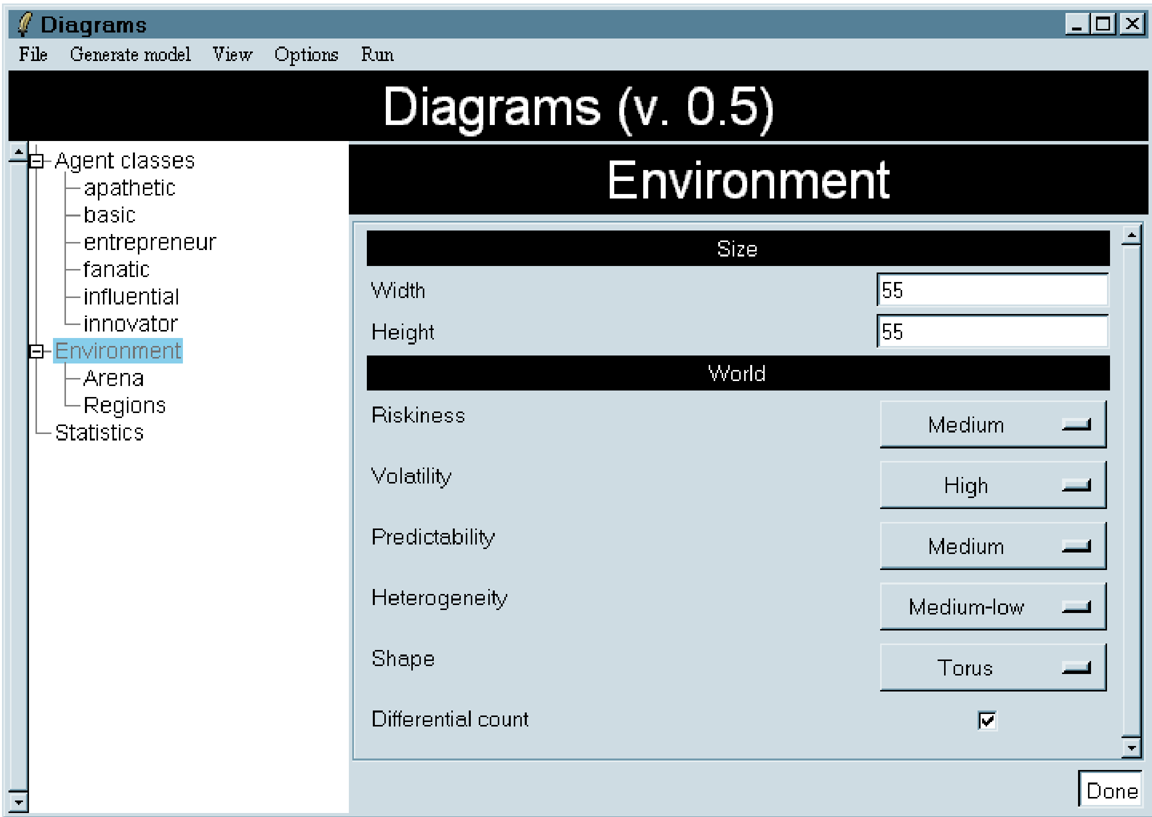
button to the right of "Create copy of this agent class." The name of the new agent class will then appear in the list under "Agent classes" on the left side of the Diagrams display. Creating a new agent class in this way does not remove any other agent class. Each default agent class remains available for use with the model being developed with the default settings assigned to it or with new settings entered by the user. The new agent class can be removed by clicking the "Invoke" button corresponding to "Destroy this agent class."

22. Once the agent class has been defined to the user's satisfaction the user can determine the presence or absence of agents of the agent class in different regions of the landscapes produced by the model being designed. This is accomplished by typing numbers from 0 to 10000 in the boxes corresponding to Region A, Region B, etc (see below). Unless regions are actually designated, the model produced will distribute agents of the particular agent class randomly and relatively evenly across the landscape.

Environment

23. When creating a model in Diagrams mode the user has the option of creating an "arena" within a larger "environment." This is particularly useful for studying the effects of external or general parametric variables on patterns of change within an arena such as a state. Clicking on the Environment button produces an array of options to set the size of the entire landscape (its width and height measured in numbers of cells in the grid) and important overall characteristics of the "World." Figure 23 shows the display opened when the Environment button is clicked.

Figure 23: Diagrams, Environment, with Selected Settings



24. Riskiness refers to the latitude of variation in biases assigned to particular identities. It can be set by clicking on the appropriate button to "Low," "Medium," or "High," depending on whether the user wants the World to be changing within a narrow range of values relevant to competing identities, within a medium range, or within a wide range. Volatility and Predictability are adjustable in the same way. Volatility refers to how rapidly bias assignments to identities are liable to change. Predictability refers to how likely it is that biases will change in a random order (maximally unpredictable) or only to ordinally adjacent values. Clicking the "Heterogeneity" button allows the user to set the size of the "spectrum" of identities available in the World (including the arena). There are five settings available under this button, including "Medium-low" and "Medium-high." The overall "Shape" of the World (the degree of wrapping present) can be set with the shape button for:

- "rectangle" (agents on extreme edges of the environment do not touch each other)"
- "cylinder" (agents on the extreme vertical edges of the environment touch each

other, but not agents along the top and bottom edges)

- “torus” (agents along all edges of the environment touch their counterparts on the opposite edge)

25. In Figure 23 we can see that the user in this case has set the width and height of the World to 55 by 55, “Riskiness” to Medium, Volatility to High, Predictability to Medium, Heterogeneity to Medium-low, and Shape to Torus. The “Differential Count” button allows the user to toggle between two general techniques agents can use to calculate identity weights pertaining to when agents change their activated identity or substitute one identity for another in their repertoires.

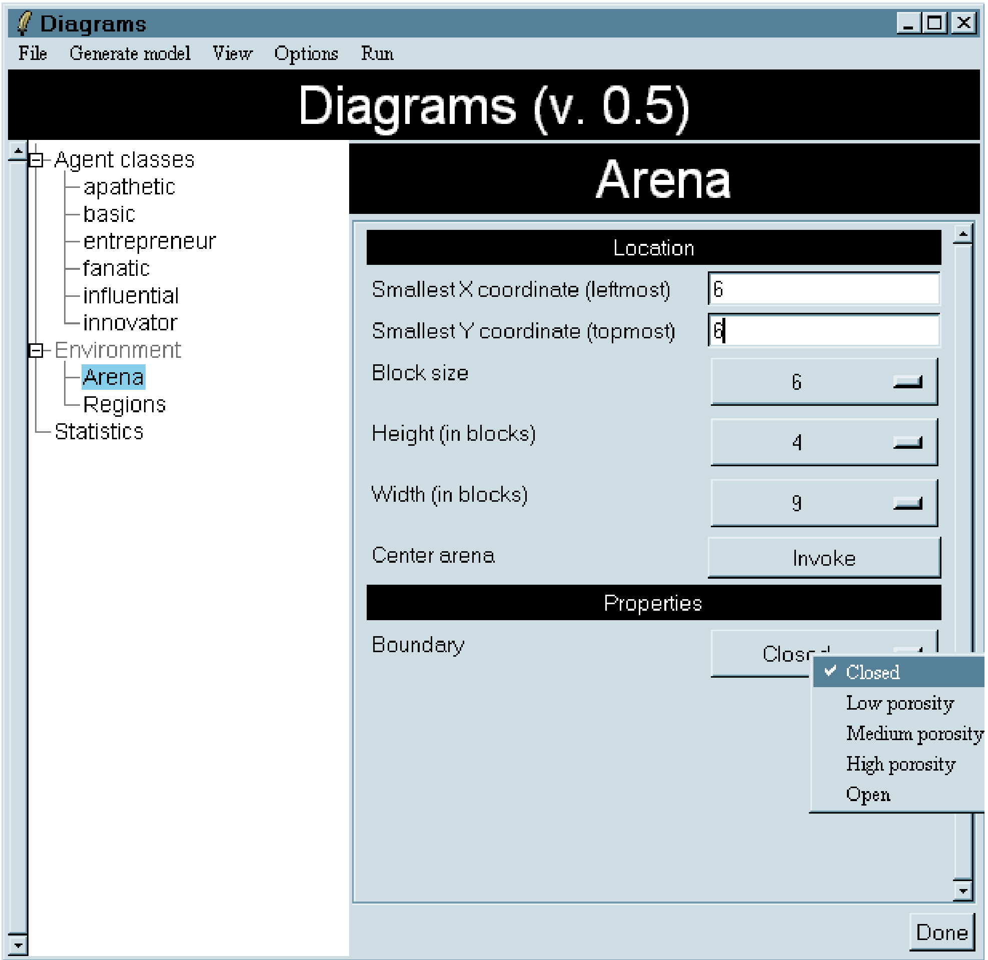
26. “Differential Count” means an agent changes in response to the most substantial impetus for change present at a particular update opportunity. If the Differential Count box is not checked, “Absolute Count” is used. This means that an agent changes in response to the first observed rule for change which is satisfied by conditions present at any update opportunity—1) rotation, 2) substitution, 3) substitution with immediate activation. We have standardly used “Differential Count.

    - *Note*: Except for Differential vs. Absolute Count, all other values established in diagrams mode can be adjusted precisely using PS-I’s own regular capabilities as explained in this manual.

Arena

27. Figure 24 shows the display that opens when the Arena button is pressed. It offers the user the opportunity to specify the location of the polity (state, village, or other “arena” to be simulated) within the environment, as well as to determine its orientation and the porosity of the border separating it from the rest of the World.

Figure 24: Diagrams, Arena with Boundary Menu Displayed



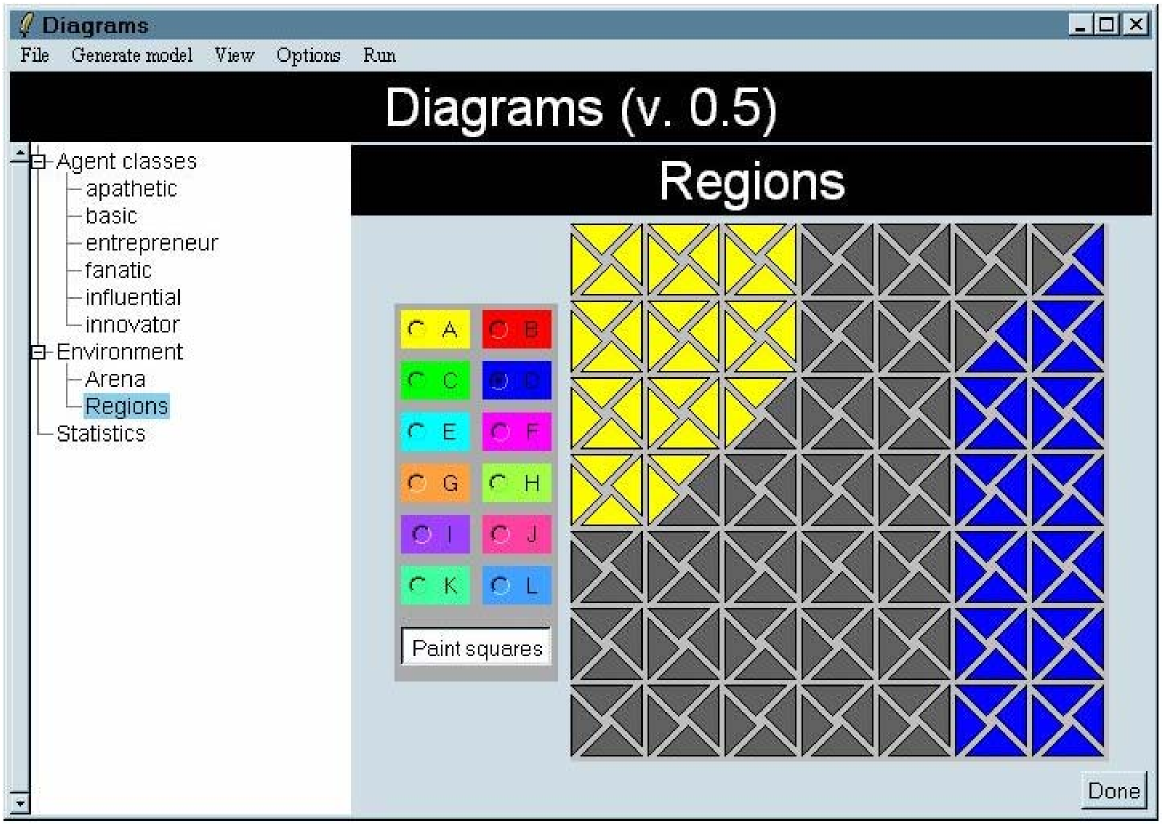
Thus in Figure 24 we see that the location of the arena within the world is fixed by the fact that its top-left corner is to be located at (6,6). Its size in regional blocks is 4 by nine indicating a rectangle that is more than twice as wide as it is high. Invoking "Center arena" will adjust the location of the arena within the world so that its center is in the center of the entire landscape (the World).

- *Note*: The "Done" button available in the displays described in Diagram mode can be clicked when the user has completed choices contained in that display. The display will then disappear. But it is not necessary to click the "Done" in order to implement changes. They are implemented as soon as the user changes to another display or clicks "Generate Model."

Regions

28. Regions are defined by clicking on “Regions” within the “Environment” section of the main Diagrams display. Thus the user defines “Region A” by painting triangles or squares with yellow. See Figure 25 in which two separate regions are designated.

Figure 25: Diagrams, Two Regions Designated—A and D



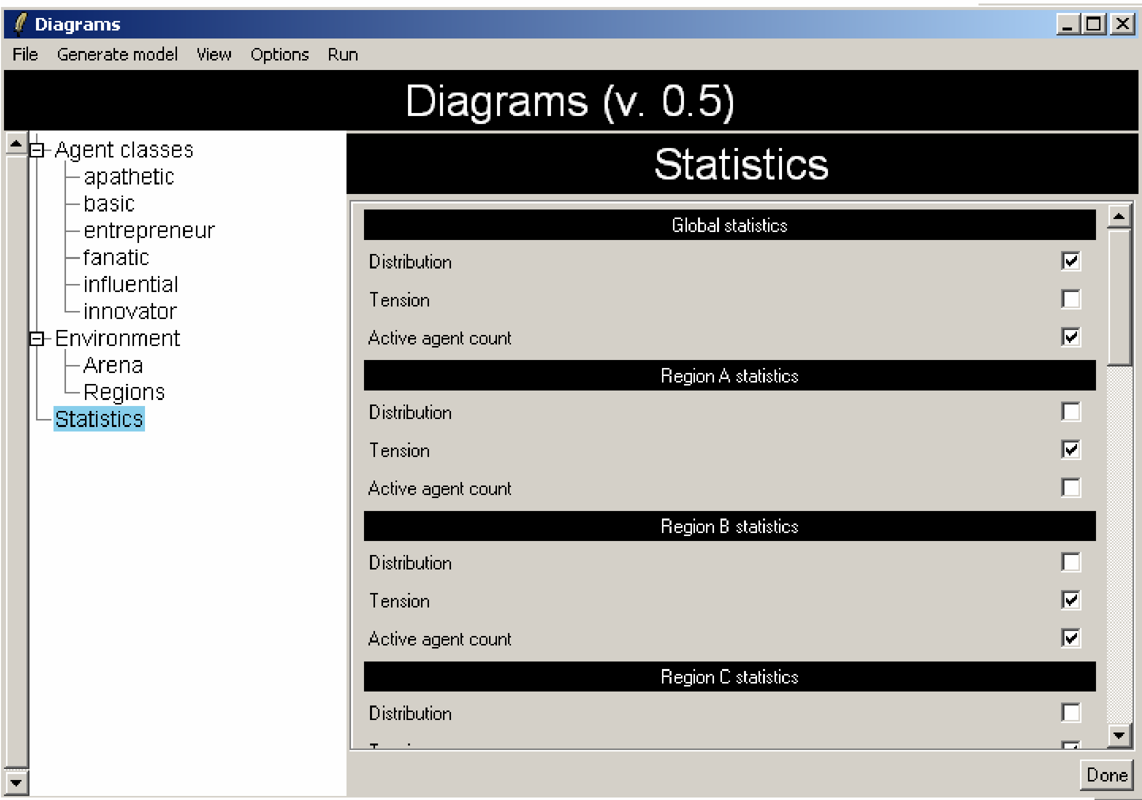
29. If the “Paint squares” button is clicked, then clicking on any triangle in a square will paint or unpaint all four triangles comprising the square. If the “Paint Squares” button is clicked again, then clicking on any triangle will paint or unpaint just the triangle that is clicked. Different regions may be defined using different colors and each class of agents can be seeded in the desired proportions in every separate region. In this way customized landscapes can be produced to correspond with various cultural, political, and institutional features of the polity or type of polity to be simulated.

Statistics

30. In Diagrams mode the user is provided with point and click opportunities to decide what statistics can be gathered by PS-I whenever the model being produced is run. Statistics can be gathered that describe the entire landscape or that describe one or more of the

particular regions used in the creation of the model. Figure 26 shows the display that appears when the "Statistics" button is clicked.

Figure 26: Diagrams, Statistics Display



31. Checking in the "Distribution box" will produce bias values, activated identity prevalence, and other information for every time step for each identity. "Global statistics" reports on the whole landscape. Different "Regional statistics" report on particular regions. Checking the "Tension box" or the "Active agent count" box, without checking the Distribution box, will result in statistics being displayed only for the total tension or for the distribution of identity activation.

Generating the Model

32. The user can view or experiment with the model under construction by clicking "Generate Model" and then using the "View" menu to click the "New Field Viewer" button. *When viewing and editing the model in this way it is important to remember that PS-I remains in Diagrams mode.* Most (though not all) of the observational and editing tools available in PS-I itself are available in Diagrams mode to inspect and evaluate landscapes viewable in Diagrams Mode. If the user wants to save the file in a form that would permit continued work on it in Diagrams mode then the File button in the Diagrams display should be clicked. If the "Save settings" button is clicked then those settings, with the name assigned by the user, will be available to be reloaded (using the "load settings" button. These "in-progress" files have a "dia" extension assigned

automatically. *However, if the model is deemed suitable, the model may be saved for future use as an mdl or snp file. However, to do so the user must return to normal PS-I mode.* This is accomplished by clicking the minimized PS-I icon, clicking the file button, and then saving either the model or the model and the snapshot by clicking either the "save edit window" or "save state" buttons.

Appendix 1

Selection and Effect Command Library

Selection Commands:

Commonly used free hand commands users can enter into the free hand box in the Selection Editor. These commands can be joined using "and" and "or" to highlight conjunctive or disjunctive sets. All commands are case sensitive. All free hand commands can be used as supplements to commands entered into the Selection Editor by typing and clicking in the appropriate line above the free hand box.

- *Note*: For ease of use, commands are represented here, not entirely abstractly, but with exemplar integers or exemplar agent classes employed. But changing an integer, from a 3 to a 5 or from a 0 to a 10, would not change the way the command is written or its effect; nor would changing an agent class name (e.g. from "entrepreneur" to "basic").

To highlight entrepreneurs (or any agent class): agentclass==[[#entrepreneur]]

To highlight inactive agents: [inactive]

To highlight immutable agents: [immutable]

To highlight agents of a certain influence: agent_influence 2 or agent_influence (3,4,6)

To highlight agents with a particular influence level (2) and activated on a particular identity (0):

agent_influence==(2) with 0 activated ID also selected

To highlight agents with influence 2 or influence 4:

agent_influence==(2) or agent_influence==(4)

To highlight agents with influence more than 1:

[influence]>1

To highlight agents with rep size of 5:

set_size([cache] intersect cache_set)==(5)

To highlight all with activated 5:

(activated([cache]))==(5)

To highlight agents subscribed on a particular identity:

[cache] intersect set (2)

To highlight agents subscribed on two particular identities:

[cache] intersect set (2,3) == set (2,3)

To highlight agents subscribed to any one of a group of identities (2,3,6,7):

[cache] intersect set (2,3,6,7)

To highlight agents with activated 5:

(activated([cache]))==(5)

To highlight different activated identities (2,10):

(activated([cache])==2) or (activated([cache])==10)

To highlight agents activated on identity 5 and having identity 2 in repertoire:

(activated([cache]))==(5) and [cache] intersect set (2))

To highlight all angry agents:

subscribed_bias_test_and_si_modified2(2,5,0) and (display_tension=>3)

Effect Commands:

A. ***Commands for Seeding with random activated:***

To seed with random activated and random filled repertoire—total number in cache = default for basic agent rep size:

make_repertoire(random_subset)

To seed with random activated and random filled rep--total number in cache = 9:

make_repertoire(random_subset(9,cache_set))

To seed with random cache of 9 Ids total excluding 14 and 15:

make_repertoire(random_subset(9,cache_set-set(14,15)))

To seed with agents having identities 4,5,6,7,9,13 in their repertoires with activated drawn at random from that set:

make_repertoire(random_element(set(4,5,6,7,9,13

B. <u>Commands to seed with specific identity activated but with specific changes to the repertoires of affected agents:</u>

To seed with agents activated on 5 and having nothing in their repertoires except 5:

repertoire(5,set(5))

To seed with agents activated 5 and 0,4,9,13,15 in repertoire:

repertoire(5,set(0,4,9,13,15))

To seed with agents activated on 2 with a total cache size of 7, and with 10 in the repertoire:

repertoire(2,set(10) union random_subset(5,cache_set-set(10)))

Apply whatever effect is specified in the "New Value" section of the Effect Tool onto 50% of agents in described rectangle (insert in the "only such agents that" window of the effect tool):

rectangle(28,28,48,48) and (rand<5000)

To seed with agents activated on 2, with a total cache size of 7, and 10 but NOT 3 in repertoire:

repertoire (2,set(10) union random_subset(5, cache_set-set(3,10)))

To seed with agents activated on 9, with a total cache size of 4 drawn from the whole spectrum.

repertoire (9,set(9) union random_subset(3, cache_set-set(9)))

To seed with agents activated on 9, with 3 in the repertoire and two others drawn from the entire spectrum.

repertoire (9,set(3) union random_subset(2, cache_set-set(9,3)))

}

To seed with agents activated on 2, with 0 in repertoire plus one of either 16,23, or 25:

repertoire(2,set(0,2)) union random_subset(1,set(16,23,25))

To seed with agents activated on 5 including 4,5,3,13,15 in repertoire, and either 0 or 11:

repertoire(5,set(5,4,13,3,15)) union random_subset(1,set(0,11))

To seed with random activated with entire spectrum in cache

make_repertoire(random_element(set([cache])))

To seed with random activated from identities already in subscription

repertoire(random_element(subscription_set([cache])), [cache])

To seed with activated on 2, already in repertoire, with all other identities remaining in repertoire.
repertoire(2,(subscription_set([cache])), [cache])

To seed with agents activated on either 0 or 11 and including in their repertoires all of 5,4,13,11,0,1 and also either 3 or 15:

repertoire(random_element(set(0,11)), set(5,4,13,11,0,1) union
random_subset(1, set(3,15)))

To seed with agents activated randomly drawn from 1,2,4 but also with 5 in their repertoires:

Repertoire(random_element(set(1,2,4)),set(1,2,4,5)

C. <u>Commands for removing or adding identities to agent repertoires</u>:

Note: The remove (unsubscribe) command will not remove an identity from an agent's repertoire if that identity is activated, nor will an identity inserted into an agent's repertoire with the insert (subscribe) command be activated.

To remove identity 4 from the repertoires of agents:

unsubscribe_identities([cache], set(4))

To insert identity 4 into the repertoires of agents:

```
subscribe_identities([cache], set(4))
```

To change the activated identities of agents by drawing only upon identities already in the repertoires of affected agents:

```
repertoire(random_element(subscription_set([cache])), [cache])
```

Appendix 2

Glossary for Model Parameters included in Model4.mdl

Parameter:

1. *bias_max*: an integer. The maximum value assignable as a bias to any particular identity.

2. *bias_min*: an integer. The minimum value assignable as a bias to any particular identity

3. *bias_volatility*: an integer from 0 to 10,000 yielding a value for this parameter of between 0 and 1. As the numerator over the denominator of 10,000 the number entered into this edit window indicates the rough probability within any particular update (conducted every other time step) that any particular identity will be eligible for a change in the bias assigned to it. Once an identity is deemed eligible for change in its bias assignment, a new bias assignment is drawn from those available. If the bias assignment produced by this procedure is the same as that already assigned to the identity then, in effect, the bias will not change. Accordingly, depending on the number of available biases (i.e. the bias "range"--see below) and the predictability setting (see below) the probability of an identity actually receiving a bias different from the bias currently assigned to it in any one update is somewhat lower than that expressed by the bias_volatility fraction.

4. *unpredictability_factor*: an integer from 0 to 10,000 yielding a value for this parameter of between 0 and 1. Set at zero, the bias assigned to an identity can be changed, but only to an available bias assignment that is not more than one, in absolute value, either less or more than the current bias assignment. As the number typed into this editing window is increased, so is the probability that a new bias assignment will result in bias that is not "near" the currently assigned bias. Set at 1 (i.e. with 10,000 typed into this editing window) a new bias has an equal probability of being assigned regardless of how near or far it is to the identity's current bias (as long, of course, as it is a bias within the "range" stipulated by the b_range parameter--see below). In other words, the larger the fraction here, the more likely a new bias assignment is to "jump" from a relatively low setting to a relatively high setting, or vice versa.

5. *permanent_set*: Under conditions stipulated with the parameter settings labeled "level1" and "level3" (see below) different types of agents can bring new identities into their repertoires, or even immediately activate on identities not in their repertoires. This can occur when identities are, for local and bias related conditions, powerfully attractive to an agent despite the fact that they are not already present in the agent's repertoire. When a new identity is taken in by an

agent, some other identity, currently present in the repertoire of the agent, is discarded.

- *Note*: The identity discarded is the identity, usually, farthest to the right in the list of identities as revealed in the Agent Viewer.

Identities listed in this identity window are "permanent," meaning once there are acquired, or if they are present in an agent's repertoire at the beginning of a run, they cannot be discarded. They remain permanently within the agent's repertoire, though not necessarily as the activated identity.

6. *unobtainable_set*: No identity listed in this edit window can be brought into the repertoire of any agent that did not begin the run of the landscape with that identity in its repertoire. Although these identities cannot be acquired as a result of interactions during the history of the landscape, agents can discard them. They are, in other words, not permanent. Nor need they be activated.

7. *bias_seed:* PS-I uses a random number generator which produces a unique stream of numbers associated with any particular number appearing in this edit window. The number appearing in this edit window is the seed for producing bias assignments for different identities over time--assignments that would be associated with a history produced by clicking the run button. By changing the number in this edit window, and clicking "set," a different stream of bias signals would be sent to the agents in the landscape as it proceeds through time, thereby producing a different history--though how much different is an "empirical" question. A separate seed, used to randomize elements of transformation, or evolution, of agents from one agent class to another, appears in the model parameter list as the "evolution_seed." (see below). Unless specific locations or complexions of agents are changed, any landscape run forward in time with the same bias and evolution seeds will reproduce exactly the same outcomes. This allows the user to introduce randomness into every aspect of the model which involves probability (by changing the seeds), but also to save the history by saving a "snapshot" of the landscape at time 0 along with the seeds associated with a particular "run." Thus it has been possible to save histories without having to save hundreds of individual snapshots (which would take enormous amounts of storage space and which would not be able to be easily displayed in dynamic form).

8. *b_range*: The integer appearing in this edit window indicates the radius of the "sight" of the agent type. B_range indicates the range of a "B"asic agent. A range of "1" indicates that the neighborhood used by every basic agent in its identity weight calculation takes into account only the agents immediately adjacent to it--that is a neighborhood comprised, in addition to itself, of 8 agents. A range of "2" would indicate that the agent type specified would take into

account 24 agents in addition to itself, i.e. all agents adjacent to itself and those adjacent to those agents.

9. *evolution_seed*: See above, "bias_seed" The number appearing in this edit window is the seed for operationalizing evolution rules that entail probabilities.

10. *Bt_sample_area*: The evolution rules developed to study conditions producing secessionism involve a rather complex set of rules derived from prevailing theories of what is necessary for secessionism to take place. For example, these rules as implemented in model 4 require agents to be activated on an identity classified as both “SI” and “OI” in order to be eligible for change into a border agent. As noted in the manual’s discussion of the Statistics Display, “SI” stands for “subordinate identity” (any identity, not the dominant identity, which is activated by at least 10% of the active agents in the landscape). “OI” stands for “opposition identity” (any identity of which it is true that no more than 20% of agents activated on that identity have the dominant identity present in their repertoires). The Bt_sample_area indicated which portion of the landscape PS-I uses to determine which identities are, in any time step, “SI” and/or “OI.”

11. *bt_transform_area*: Most of the evolution rules governing change from one agent type to another apply to all agents of in the landscape. Sometimes it may be desirable to apply the rules only to a portion of the agents—those located in one part of the landscape. In Model4 the rules allowing other types of agents to transform into border agents under certain circumstances (useful for studying secessionism, for example) are only to be applied to agents located in the area specified in this edit window.

12. *BC*: The probability that an agent otherwise eligible to be transformed into a border agent will do so at any particular updating time step. As elsewhere in PS-I this probability is expressed as a fraction with 10,000 as its denominator.

13. *HI*: Herfindahl Index. Traditionally employed to describe the extent of concentration in an industry or market. Here the index, as a number between zero and one, describes the amount of concentration in the shares of the landscape occupied by agents activated on different identities. The index is calculated as equal to the sum of the squares of the shares of the activated agents in the landscape. Thus the higher the number the more concentrated is pattern of identity activation, with a small number of identities activated by the overwhelming majority of agents. If all agents were activated on one and only one identity the Herfindahl Index would be 1. The number as it appears in this window changes as the landscape moves forward through time. It is useful in itself as a measurement tool and can play a role in the calculation of other important values.

14. *ATL*: Average Tension of a Landscape the total tension in a landscape divided by the number of active agents. Tension refers to the number of encounters an agent

has in its neighborhood with agents activated on an identity different from that on which it is activated.

15. *ATI:* Average Tension for each identity presently activated in the landscape. Each identity in this edit window appears with the average tension experienced by those agents currently activated on that identity.

16. *DI*: Dominant Identity. The identity activated by more agents than any other.

17. *SI*: Subordinate Identity. Those identities classified as "subordinate." In Model4 an SI is any identity activated by at least 10% of the active agents in a landscape but which is not the DI.

18. *OI*: Opposition Identity. Those identities classified as "Opposition." In Model4 an OI is any identity with regard to which no more than 20% of the agents activated on that identity have within their repertoires (subscriptions) the identity which, at that time step, is the dominant identity in the landscape.

19. *b_level1*: For a "B" (basic) agent, when to add an identity to its repertoire and discard one present within it but not activated (when to "substitute" an identity for an unactivated identity in the repertoire of an agent). The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an unactivated identity in a basic agent's repertoire must exceed the identity weight of the activated identity of that agent in order for the unactivated identity to be rotated into the activated position.

20. *B_level2*: For a "B" (basic) agent, when to replace the identity activated by a basic agent with an unactivated identity in the agent's repertoire (when to "rotate" identities). The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an identity not present in a basic agent's repertoire must exceed the identity weight of the activated identity of that agent in order for it to be inserted into the repertoire of that agent and for an unactivated identity to be discarded.

21. *B_level3*: For a "B" (basic) agent, when to substitute an identity <u>not</u> in an agent's repertoire for the identity activated by that agent. The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an identity not included within a basic agent's repertoire must exceed the identity weight of the activated identity of that agent in order for that "foreign" identity to be immediately activated by the agent.

22. *B_influence*: For a "B" (basic) agent, the influence expressed by a single agent on each of its neighbors in support of its currently activated identity. Identity weight calculations by an agent accord one "point" per identity for every agent in its neighborhood activated on that identity, multiplied by the "influence" of the agent activating that identity. So if there are two agents in agent "X's" neighborhood

activated on identity 3, and each is a basic agent (i.e. an agent with an influence level of 1), then agent X counts 2 points from those [(1*1=1)+(1*1=1)}agents as a contribution to its calculation of the identity weight of identity 3.

23. *e_range*: The integer appearing in this edit window indicates the radius of the "sight" of the agent type. E_range indicates the range of an "E"ntrepreneur agent. A range of "2" indicates that in model4, as here stipulated, the neighborhood used by every entrepreneur agent in its identity weight calculations takes into account not only the agents immediately adjacent to it--that is a neighborhood comprised, in addition to itself, of 8 agents, but also the 16 outside agents immediately adjacent to them. (see above under "b_range."

24. *e_level1*: For an "E"ntrepreneur agent, when to add an identity to its repertoire and discard one present within it but not activated (when to "substitute" an identity for an unactivated identity in the repertoire of an agent). The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an unactivated identity in a basic agent's repertoire must exceed the identity weight of the activated identity of that agent in order for the unactivated identity to be rotated into the activated position.

25. *e_level2*: For an "E"ntrepreneur agent, when to replace the identity activated by a basic agent with an unactivated identity in the agent's repertoire (when to "rotate" identities). The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an identity not present in a basic agent's repertoire must exceed the identity weight of the activated identity of that agent in order for it to be inserted into the repertoire of that agent and for an unactivated identity to be discarded.

26. *e_level3*: For an "E"ntrepreneur agent, when to substitute an identity <u>not</u> in an agent's repertoire for the identity activated by that agent. The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an identity not included within a basic agent's repertoire must exceed the identity weight of the activated identity of that agent in order for that "foreign" identity to be immediately activated by the agent.

27. *e_influence*: For an "E"ntrepreneur agent, the influence expressed by a single agent on each of its neighbors in support of its currently activated identity. Identity weight calculations by an agent accord one "point" per identity for every agent in its neighborhood activated on that identity, multiplied by the "influence" of the agent activating that identity. So if there are two agents in agent "X's" neighborhood activated on identity 3, and each is an entrepreneur agent (i.e. an agent with an influence level of 2), then agent X counts 4 points from those [(1*2=2)+(1*2=2)}agents as a contribution to its calculation of the identity weight of identity 3.

28. *i_range*: The integer appearing in this edit window indicates the radius of the "sight" of the agent type. i_range indicates the range of an "I"nnovator agent. A range of "1" indicates that in model4, as here stipulated, the neighborhood used by every innovator agent in its identity weight calculations takes into account only the agents immediately adjacent to it. (see above under "b_range.")

29. *i_level1*: For an "I"nnovator agent, when to add an identity to its repertoire and discard one present within it but not activated (when to "substitute" an identity for an unactivated identity in the repertoire of an agent). The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an unactivated identity in a basic agent's repertoire must exceed the identity weight of the activated identity of that agent in order for the unactivated identity to be rotated into the activated position.

30. *i_level2*: For an "I"nnovator agent, when to replace the identity activated by a basic agent with an unactivated identity in the agent's repertoire (when to "rotate" identities). The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an identity not present in an innovator agent's repertoire must exceed the identity weight of the activated identity of that agent in order for it to be inserted into the repertoire of that agent and for an unactivated identity to be discarded.

31. *i_level3*: For an "I"nnovator agent, when to substitute an identity <u>not</u> in an agent's repertoire for the identity activated by that agent. The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an identity not included within an innovator agent's repertoire must exceed the identity weight of the activated identity of that agent in order for that "foreign" identity to be immediately activated by the agent.

32. *i_influence*: For an "I"nnovator agent, the influence expressed by a single agent on each of its neighbors in support of its currently activated identity. Identity weight calculations by an agent accord one "point" per identity for every agent in its neighborhood activated on that identity, multiplied by the "influence" of the agent activating that identity. So if there are two agents in agent "X's" neighborhood activated on identity 3, and each is an innovator agent (i.e. an agent with an influence level of 1), then agent X counts 2 points from those agents [(1*1=1)+(1*1=1)] as a contribution to its calculation of the identity weight of identity 3.

33. *F_influence*: For an "F"anatic agent. A fanatic agent is conceived as vehemently and persuasively expressing its activated identity, but not monitoring or being influenced by signals from its neighborhood or elsewhere as to the value of activation on that identity relative to other possible identities that might be activated. The integer in this edit window is the influence level of a fanatic agent--3 in this version of model4. So if there are two agents in agent "X's"

neighborhood activated on identity 3, and each is a fanatic agent (i.e. an agent with an influence level of 3), then agent X counts 6 points from those agents [(1*3=3)+(1*3=3)] as a contribution to its calculation of the identity weight of identity 3.

34. *a_range*: The integer appearing in this edit window indicates the radius of the "sight" of the agent type. a_range indicates the range of an "A"pathetic agent. An apathetic agent is conceived as an agent that monitors its environment and changes its activation in response to signals it receives but does not influence those around it. Its activated identity is not monitored by its neighbors or counted in their calculations of identity weights. Apathetic agents are "inactive"--as if their influence were 0. A range of "1" indicates that in model4, as here stipulated, the neighborhood used by every apathetic agent in its identity weight calculations takes into account only the agents immediately adjacent to it. (see above under "b_range.")

35. *a_level1*: For an "A"pathetic agent, when to add an identity to its repertoire and discard one present within it but not activated (when to "substitute" an identity for an unactivated identity in the repertoire of an agent). The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an unactivated identity in an apathetic agent's repertoire must exceed the identity weight of the activated identity of that agent in order for the unactivated identity to be rotated into the activated position.

36. *a_level2*: For an "A"pathetic agent, when to replace the identity activated by an apathetic agent with an unactivated identity in the agent's repertoire (when to "rotate" identities). The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an identity not present in an apathetic agent's repertoire must exceed the identity weight of the activated identity of that agent in order for it to be inserted into the repertoire of that agent and for an unactivated identity to be discarded.

37. *i_level3:* For an "A"pathetic agent, when to substitute an identity <u>not</u> in an agent's repertoire for the identity activated by that agent. The integer in the edit window represents, for this model, the minimum amount measured in identity weight by which an identity not included within an apathetic agent's repertoire must exceed the identity weight of the activated identity of that agent in order for that "foreign" identity to be immediately activated by the agent.

38. *self_influence*: The integer in this edit window indicates the multiple used to determine the contribution which an agent's own activated identity makes to its identity weight calculation. A zero written here would mean that agents would not count their own activated identity in their identity weight calculations. A two written here would mean that agents count their own activated identity as "2" in their identity weight calculations.

Appendix III

Designing Landscapes and Running Experiments with Scripts

Dan Miodownik/ August 2005

Introduction

What is a script?

A script is a text file that permits you, a PS-I user, to design landscapes and models from scratch and implement experimental protocols using pre-designed simulation environments. Once loaded by Ps-I a script can instruct the program to do one or a combination of many things:

- load particular files (snapshots or model);
- Set distributions of agents with specific characteristics on the grid;
- Select and change characteristics of part (or all) agents on the grid.
- Set value(s) or restrict values of parameter defined in the model specification file;
- Run a model for particular periods of time;
- Implement changes during a rendition;
- Collect and save desired statistical information for a specific time (or for the entire an entire run).

Why should one use a script?

Scripts help produce hundreds and even thousands of dynamic runs of a pre-designed snapshot or model without the user having to guide the process step by step. One can think of scripts as the tool for instructing PS-I, a tool that permits users to conceive of and employ an extensive experimental design without the need to stop and restart experiments in order to set different parameter values to independent and interaction effects. All necessary treatment conditions can be implemented, in other words, by being specified within script.

The scrip syntax contains information and commands executed one at a time by PS-I. There is almost an infinite number of ways to write and modify a script. The most important thing is to remember that a script serves as a bridge between the user and PS-I.

The most useful kinds of commands included in scripts are those used to systematically implement changes to one or several parameters, to change a landscape at some predetermined point or point during a dynamic run, or to design landscapes with particular characteristics without needing to "manually" make adjustments in the parameter settings or model specification file of particular models or snapshots.

This manual is designed to enable users to take advantage of the scripting techniques available in PS-I. This manual assumes that you have some experience using PS-I, and well familiar with the language, and most of the jargon, related to PS-I. I will not, therefore spend time and space on definitions and reviews of key concepts. You can

find all of those in the official PS-I manual available at http://www.polisci.upenn.edu/abir. In addition, notice that all the examples included in this scripting manual are based on model4.mdl. Recall that this model, that is available with PS-I installation kit, is the central model used as an illustration of the topics covered in the toolkit's manual.

Running a Simple Script

Let us begin with an example of a simple generic (and most frequently used) script. Later, I will review ways to modify the script to fit more specific experimental needs. Consider the following syntax:

```
set x 100
for {set i 0} {$i < $x} {incr i 1} {
load_file "Model4.mdl"
file delete "model4(baseline)_$i.csv"
set_filename "model4(baseline)_$i.csv"
for {set k 0} {$k < 100} {incr k 1} {step}
}
```

Each line in the script contains information ordering PS-I to behave in a certain way. This specific script tells PS-I to load a model (model4.mdl), run it for several time steps, save statistical information as a text file (comma delimited, *.CSV), save a snapshot representing the spatial configuration of the agents at the last time step, and repeat the same process several time.
Let us now turn to the script itself and review the command lines it includes.

Script Name: Run_Model4.scp

The first line sets the number of times (x) to repeat the loop (x=10)

```
set x 10
```

The second line tells PS-I to execute subsequent commands (following the open curly
bracket), as long as (i) [the number of the current repetition] is smaller than x.

```
for {set i 0} {$i < $x} {incr i 1} {
```

 # NOTE: The second line tells PS-I to implement a set of specified commands, and repeat that # operation several times. PS-I will implement all the commands that appear between the open # curly bracket and the close curly bracket (the brackets are painted in red).

The third line tells PS-I to load model4.mdl

load_file "model4.mdl"

The fourth line tells PS-I to delete any *.csv file carrying a specific name.

file delete "model4(baseline)_$i.csv"

 # NOTE: It is recommended that you use filenames that provide some useful and
recognizable information. Typically the file name is composed of two parts; one
indicating the name of the snapshot/model and the experimental condition used to
#generate the data (model4(baseline)) and another part (_$i) that indicates the number of
the specific run. Remember that unless you delete old files and provide unique names
for each new data file, PS-I will store all the data in the same data file.

The fifth line of command tells the program to store statistical data as *.csv files carrying a
specific name.

set_filename "model4(baseline)_$i.csv"

NOTE: PS-I will report values for each parameter specified in the model specification
file. You need, therefore, to make sure that the statistics section of the model
specification file contains syntax telling PS-I to collect information on the particular
parameter(s) you may be interested with.

The sixth line set a length for the run, telling PS-I to step forward in time as long as (k) is
smaller than 10.

for {set k 0} {$k < 10} {incr k 1} {step}

The seventh line orders PS-I to save a snapshot of the last time step.

save_state_to_file "model4(baseline)(t=10)_$i.snp"

The eighth line ends the loop and orders PS-I to attempt to begin a new one.

}

Using Scripts to apply effects:

Next, let us consider syntax you may want to use in order to apply effects either upon loading a model or during a dynamic run. To make sure that PS-I recognizes the syntax one needs to include in the apply_effect command three pieces of information: (a) the name of the field to apply the command to (i.e. landscape), (b) an expression that defines the area of the effect, and (c) the list of attributes to changed (the effect itself).

In principle one can use scripts to change any of the attributes that can be affected by using PS-I's effect tool. Below you can find a long list with examples of typical effect commands. You may modify them and embed them within a script. Alternatively, you can use PS-I's effect tool to generate the specific syntactic language for effects you want to implement. To do so specify the effect using the effect tool, press where it *says copy effect command to clipboard* and pasting the syntax into the appropriate location in the script (see below for two examples of scripts with embedded effects commands).

Examples of commonly used apply_effect syntax:

To seed the entire of a landscape with basic agents:

*apply_effect "landscape" "(rand<10000)" {
-1 "basic"}*

To seed the entire area of the landscape with basic agents that are immutable:

*apply_effect "landscape" "(rand<10000)" {
"immutable" "true" -1 "basic" }*

To seed 50% of the agents with an activated identity 2 and repertoire that includes identity 0:

*apply_effect "landscape" "(rand<5000)" {
"cache" "repertoire(2,set(2,0))"}*

NOTE: A more extensive list of effect commands that one can use to seed agents'
repertoires are available on pages 2-5 of appendix I in the PS-I manual.

To seed the entire area of the landscape with basic agents that are immutable and
inactive:

*apply_effect "landscape" "(rand<10000)" {
"immutable" "true" "inactive" "true" -1 "basic" }*

To seed the entire area of the landscape with basic agents that are immutable, inactive
and activated on identity 1 with identities 2, and 3 in their repertoire:

```
apply_effect "landscape" "(rand<10000)" {
    "immutable" "true"  "inactive" "true"  -1 "basic" "cache"
    "repertoire(1, set(1,2,3))"}
```

To seed a restricted area with basic agents that are immutable, inactive and activated on
identity 1with identities 2, and 3 in their repertoire.

```
apply_effect "landscape" "(rectangle(10, 10, 15, 15)) and (rand<10000)" {
    "immutable" "true"  "inactive" "true"  -1 "basic" "cache"
    "repertoire(1, set(1,2,3))"}
```

To seed 50% of a restricted area with basic agents that are immutable, inactive and
activated on identity 1 with identities 2, and 3 in their repertoire:

```
apply_effect "landscape" "(rectangle(10, 10, 15, 15)) and (rand<5000)" {
     "immutable" "true"  "inactive" "true"  -1 "basic" "cache"
      "repertoire(1, set(1,2,3))"}
```

To demarcate a 50 by 50 grid with an array of border agents:

```
apply_effect "landscape" "(((rectangle(0, 0, 49, 0) or rectangle(49, 1, 49, 49))
or  rectangle(1, 49, 48, 49)) or rectangle(0, 1, 0, 49))" {
      -1 "border"}
```

To set seed basic agents in a specific area as immutable, and with activated identity 1
and identity 2 in the repertoire:[1]

```
 apply_effect "landscape" "(chk_basic==1 and rectangle(10,10,15,15))  {
      "immutable"  "1" "cache"  "repertoire(1,set(1,2))"}
```

To seed a specific area with different percentages of agents from three agent-classes,
each with its own unique identity repertoire: basic (60%), entrepreneur (20%), and
apathetic (20%).

```
apply_effect "landscape" "rectangle(10,10,20,20) and (rand<10000)"{
        -1 "basic" "cache" "repertoire(1, set(1,0,2))"}
apply_effect "landscape" "rectangle(10,10,20,20) and (rand<2500)" {
        -1 "entrepreneur" "cache" "repertoire(3, set(1,3,2))"}
apply_effect "landscape" "rectangle(10,10,20,20) and (rand<2000)" {
```

[1] For reasons I explained above this requires to insert a new routine into the model specification file. Make sure to add the following to text to the model specification file, submit and save.

```
routine 'chk_basic' composite
 comment 'A routine that checks if an agent is a basic agent'
 code "(agentclass==[[#basic]])"
 end
```

-1 "apathetic" "cache" "repertoire(5 set(1,4,5))"}

Embedding apply effect commands inside a simple script:

As I have already mentioned, you can embed within a script syntax telling PS-I to apply some effect. You may decide that you want to execute the command immediately upon loading a model or at some later point during the dynamic run.

Consider the following examples (a modification of the simple script I presented above used before). The script in the first example executes the command immediately after it loads the model.

To seed half of a rectangle array in the center of the model with apathetic agents with
an identity repertoire composed of identities 1 and 0, run it for 50 time steps, and loop
10 times.

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "model4.mdl"
apply_effect "landscape" "(rectangle(23, 23, 35, 35) and
(rand<5000)" {"cache  "repertoire(1, set(1,0))"  -1 "apathetic" }
file delete "model4(apathy)_$i.csv"
set_filename "model4(apathy)_$i.csv"
for {set k 0} {$k < 50} {incr k 1} {step}
}
```

The script in the second example orders PS-I to apply the same effect after 40 time steps.

To run a model for 40 time steps and then seed half of a rectangle array in the center
with apathetic agents with an identity repertoire composed of identities 1 and 0, run it
for 10 additional time steps, and loop 10 times.

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "model4.mdl"
file delete "model4(apathy)_$i.csv"
set_filename "model4(apathy)_$i.csv"
for {set k 0} {$k < 40} {incr k 1} {step}
apply_effect "landscape" "(rectangle(23, 23, 35, 35) and
(rand<5000)" {"cache  "repertoire(1, set(1,0))"  -1 "apathetic" }
for {set k 41} {$k < 50} {incr k 1} {step}
}
```

Using Scripts to set parameter values:

Let us now consider syntax you may want in order to determine set values for routines and parameters included in the model, and/or change these values during a dynamic run of the model. In order to do so, you may want to consider several methods and techniques to adjust the *set_routine_code_by_name* syntax to your particular needs. The appropriate syntax requires that you specify two pieces of information (a) the routine name (e.g. *b_range* – the routine defining the sight radius of a basic agents), and (b) the parameter value.

Examples of commonly used set_routine_code_by_name syntax:

(1) *<u>Bias Range</u>*

To set maximum bias value to 1:

set_routine_code_by_name "bias_max" "1"

To increase the maximum bias after the 50th time step:

set_routine_code_by_name "bias_max" "(time>50) ? 3 :1"

To set the maximum bias value between the 50th and the 79th time steps to 3, otherwise
set it to 1:

set_routine_code_by_name "bias_max" "((time>50) and (time<80)) ? 3 :1"

NOTE: In order to set/change/restrict the minimum bias values juts replace the routine
name "bias_max" with "bias_min".

to set a random bias seed:

set bias_seed [call_rand]
set_routine_code_by_name "bias_seed" "$bias_seed"

(2) <u>Bias Volatility</u>

To set the bias volatility to 5%:

set_routine_code_by_name "bias_volatility" "500"

To set a high bias volatility (5000) during the first 8 time steps, and then change it to
500:

set_routine_code_by_name "bias_volatility" "(time<9) ? 5000 :500"

To set a high bias volatility (5000) between the 10th and the 19th time steps, otherwise
set it to 500:

set_routine_code_by_name "bias_volatility" "((time>9) and (time<20)) ? 5000 :500"

NOTE: In the examples below I offer syntax that would change the settings of basic
agents. To change values of any other agent-class simply replace the basic agent
specific code (e.g. b_range, b_level1, b_influence etc.) with the code appropriate to the
agent-class you wish to affect (i..e.. for entrepreneurs: e_range, e_influence etc.).

(3) <u>Sight Radius</u>

To set the sight radius of basic agents to 1:

set_routine_code_by_name "b_range" "1"

To set the sight radius of basic agents in a specific area to 2, otherwise set it to 1:

set_routine_code_by_name "b_range" "rectangle(1,1,10,10) ? 2 : 1"

To change the sight radius of basic agents from 1 to 2 after the 30th time step:

set_routine_code_by_name "b_range" "(time>30) ? 2 :1"

To change the sight radius of basic agents in a specific area from 1 to 2 after the 30th
time step:

set_routine_code_by_name "b_range" "(rectangle(1,1,10,10) and (time>30)) ? 2 :1"

To change the sight radius of basic agents in a specific area from 1 to 2 after the 30th
time step and change it back to 1 after the 44th time step:

```
set_routine_code_by_name "b_range" "(((time>30) and (time<45)) and
    rectangle(1,1,10,10))? 2 :1"
```

To set the sight radius of basic agents to any value 1 and 4:

```
set b_range  [expr round(1.0+[call_rand]*3.0/32767)]
set_routine_code_by_name "b_range" "$b_range"
```

NOTE: The value of *b_range* is calculated by calling a random number [call_rand],
multiplying it by 3, dividing it by 32767[2], adding the product to 1, and finally rounding
the expression. The result is a number between 1 (if the random number approximates
0) and 4 (if the random number approximates 32767). The value (*$b_range*) generated
in the expression is set as the permanent value of the *b_range* routine.[3]

To set the sight radius of basic agents to either in a specific area to either 2 or 3,
otherwise set it to 1:

```
set rangeb  [expr round(2.0+[call_rand]*1.0/32767)]
set_routine_code_by_name "b_range" "rectangle(1,1,10,10) ? $rangeb  : 1"
```

To set the sight radius of basic agents to the same sight radius as entrepreneurs:

```
set_routine_code_by_name "b_range" "e_range"
```

To set the sight radius of basic agents in a specific area to the same as entrepreneurs,
otherwise set it to 1:

```
set_routine_code_by_name "b_influence" "rectangle(1,1,10,10) ? e_range : 1"
```

(3) Triggers: Rotation Trigger

To set the rotation triggers of basic agents to 2:

```
set_routine_code_by_name "b_level2" "2"
```

[2] The built in random number generator draws numbers between 0 and 32767. The product of [call_rand]/32767 would be, therefore, a number between 0 and 1.

[3] One should use similar syntax to set random values for any routine included in the model. Make sure to adjust the language, and the expression to reflect the routine you wish to affect, and to set appropriate upper and lower margins for the random values. More examples are available in the next few pages.

To set the rotation triggers of basic agents in a specific area to 4, otherwise set it to 2.

```
set_routine_code_by_name "b_level2" "rectangle(1,1,10,10) ? 4: 2"
```

To change the rotation trigger of basic agents to 4 after the 30th time step:

```
set_routine_code_by_name "b_level2" "(time>30) ? 4 :2"
```

To change the rotation trigger of basic agents in a specific area to 4 after the 30th time
step, otherwise set it to 2:

```
set_routine_code_by_name "b_level2" "(rectangle(1,1,10,10) and (time>30)) ? 4 :2"
```

To change the rotation trigger of basic agents in a specific area from 2 to 4 after the
100th time step and change it back to 2 after the 149th time step:

```
set_routine_code_by_name "b_level2" "(((time>100) and (time<150)) and
        rectangle(1,1,10,10))? 4 :2"
```

To set the rotation trigger of basic agents to any value between 0 and 8:

```
set b_level2  [expr round(0.0+[call_rand]*8.0/32767)]
set_routine_code_by_name "b_level2" "$b_level2"
```

To set the rotation trigger of basic agents in a specific area to any value between 4 and
8, otherwise set it to 2:

```
set rotationb  [expr round(4.0+[call_rand]*4.0/32767)]
set_routine_code_by_name "b_level2" "rectangle(1,1,10,10) ? $rotationb  : 2"
```

To set the rotation trigger of basic agents in proportion to 50% of their neighborhood
size:

```
set  b_level2  [expr ((((2*b_range+1)*(2*b_range+1) )-1)/2)]
set_routine_code_by_name "b_level2" "$b_level2"
```

(4) Influence level

To set the influence level of basic agents to 1:

```
set_routine_code_by_name "b_influence" "1"
```

To set the influence level of basic agents in a specific area to 2, otherwise set it to 1:

```
set_routine_code_by_name "b_influence" "rectangle(1,1,10,10) ? 2 : 1"
```

To change the influence level of basic agents from 1 to 2 after the 30th time step:

```
set_routine_code_by_name "b_influence" "(time>30) ? 2 :1"
```

To change the influence level of basic agents in a specific area from 1 to 2 after the 30th
time step:

```
set_routine_code_by_name "b_influence " "(rectangle(1,1,10,10) and (time>30)) ? 2 :1"
```

To change the influence level of basic agents in a specific area from 1 to 2 after the 30th
time step and change it back to 1 after the 44th time step:

```
set_routine_code_by_name "b_influence" "(((time>30) and (time<45)) and
        rectangle(1,1,10,10))? 2 :1"
```

To set the influence level of basic agents to any value between 0 and 2:

```
set b_influence  [expr round(0.0+[call_rand]*2.0/32767)]
set_routine_code_by_name " b_influence" "$ b_influence"
```

To set the influence level of basic agents influence in a specific area to either 0 or 1,
otherwise set it to 2:

```
set influenceb  [expr round(0.0+[call_rand]*1.0/32767)]
set_routine_code_by_name "b_influence" "rectangle(1,1,10,10) ? $influenceb  : 2"
```

To set the influence level of basic agents to the same influence level as entrepreneurs:

```
set_routine_code_by_name "b_influence" "e_influence"
```

To set the influence level of basic agents in a specific area to the same level as
entrepreneurs, otherwise set it to 1:

```
set_routine_code_by_name "b_influence" "rectangle(1,1,10,10) ? e_influence : 1"
```

Embedding set_routine commands inside a simple script:

Set routine commands can be used within a script to set parameter values. You may use these commands in at least three ways: (1) to set a value as soon as PS-I loads a new model, (b) to set, as a value, a syntactic argument that contains conditions under which the parameter value will change, and (c) set values run the model for a specified period, set new values and coninue to run the model.

Consider the following examples (once again I modify the simple script discussed above). For illustration I chose to set values of the bias maximum (you should apply the similar logic to all other routines). The first script orders PS-I to load a model, and set a value that remains unchanged until the end of the run.

```
# To set the maximum bias to 3, run the model for 50 time steps, and loop 10 times:

set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "model4.mdl"
set_routine_code_by_name "bias_max" "3"
file delete "model4(apathy)_$i.csv"
set_filename "model4(apathy)_$i.csv"
for {set k 0} {$k < 50} {incr k 1} {step}
}
```

```
# To set the maximum bias to 3 during the first 40 time steps, increase bias maximum to
# 5 during  10 additional time steps, and loop 10 times:

set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "model4.mdl"
set_routine_code_by_name "bias_max" "3"
file delete "model4(apathy)_$i.csv"
set_filename "model4(apathy)_$i.csv"
for {set k 0} {$k < 40} {incr k 1} {step}
set_routine_code_by_name "bias_max" "5"
for {set k 41} {$k < 40} {incr k 1} {step}
}
```

Alternatively, in the third and last example I combine information about the desired bias maximum during the run into one syntactic argument.

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "model4.mdl"
set_routine_code_by_name "bias_max" "(time>40) ? 5 : 3"
file delete "model4(apathy)_$i.csv"
set_filename "model4(apathy)_$i.csv"
for {set k 0} {$k < 50} {incr k 1} {step}
}
```

Combining effects and routines to design a landscape:

Up until this point you have learned how to write a simple script that tells PS-I to load a model, run it forward in time, save statistics and repeat the same operation over and over. In addition, you have seen examples of syntactic arguments that you can embed in the script in order to apply effects and set values for any of the routines available in the model specification file. In this section I show how you can employ the knowledge you gained so far to write a script that will produce a snapshot, you may later use as the foundation for your experimental work.

Characteristics of a landscape we wish to design

Let us begin by designing a landscape.

- Imagine a 50*50 grid surrounded by a closed border.
- You want to populate the grid with agents belonging to one of three agent-classes: basic, entrepreneur, apathetic.
- Each of these agent-classes is defined by the following parameters:

	Basic	*Entrepreneur*	*Apathetic*
Sight_Radius (range)	1	2	3
Rotatio Trigger (level2)	2	1	4
Substitution (level1)	5	4	8
Sub_Act (level3)	7	6	12
Influence Level	1	2	0

- You wish to divide the grid into three rectangles of equal size labeled: Left, Center, and Right. Each region is to contain 768 cells.
- With respect to agents belonging to different agent-classes you want each rectangle to be populated by the following distributions of agents in different agent-classes:

Left: Basic 60%, Entrepreneur 20%, Apathetic 20%.
Center: Basic 20%, Entrepreneur 60%, Apathetic 20%.
Right: Basic 20%, Entrepreneur 20%, Apathetic 60%.

- The distribution of identity repertoires in the grid is as follows:

Left	50% of all the agents activated on the red identity with green and blue in their repertoire, and 50% are activated on the green identity with blue in the repertoire.
Center	50% of the agents are activated on blue with red in the repertoire, and 50% are activated on green with blue and red in the repertoire.
Right	50% are activated on red with green and blue in the repertoire

and 50% are activated on blue with green in their repertoire.

Translating conditions and distributions to a script

To produce a snapshot with these distributions and parameters use the script described below:[4] The script loads model4 and uses apply_effect and set_routine_code_by_name to seed distributions of agents into the grid with specific combinations of parameters and identity repertoires. We now, then, ready to take a closer look at the commands included in the script.[5]

#Script Name: ModelWith3AC.scp

To load model4:

load_file "model4.mdl"

To surround the grid with an array of border agents:

apply_effect "landscape" "(((rectangle(0, 0, 49, 0) or rectangle(49, 1, 49, 49)) or rectangle(1, 49, 48, 49)) or rectangle(0, 1, 0, 49))" { -1 "border"}

To seed the left region with basic (60%), entrepreneur (20%), and apathetic (20%)
agents:

apply_effect "landscape" "rectangle(1,1,16,48) and (rand<10000)"{ -1 "basic"}
apply_effect "landscape" "rectangle(1,1,16,48) and (rand<2500)" {
-1 "entrepreneur"}
apply_effect "landscape" "rectangle(1,1,16,48) and (rand<2000)" { -1 "apathetic"}

NOTE: Using rand results in selecting of a number of agents approximating the desired
percentage (see manual). Notice that PS-I seeds the entire region (100% or 10000) with
basic agents, then seeds the region again with approximately 25% (2500) entrepreneur
agents, and finally seeds the entire region again with 20% (2000) apathetic agents, resulting
in the desired distribution of 60%, 20%, 20%. It is important to remember that one must
seed the entire space first in order to guarantee that all the cells will be inhibited.

[4] Comments following a number sing (#) intend to explain what is accomplished with each command or block of commands. The entire text can be copied and pasted into a text editor, saved with a script file (a text file with *.scp as filename type) and loaded by PS-I as it is. PS-I will ignore text that begins with #):
[5] The script is also available in the glossary of scripts below. It is labeled: ModelWith3AC.scp If you wish to run it, just copy and paste the text into a notepad, and save the file with the *.scp extension. To run it, simply load it using with PS-I. Make sure that model4.mdl is available in the same directory.

To Seed the agents in the left region with the desired identity combinations:

apply_effect "landscape" "rectangle(1,1,16,48) and (rand<10000)" {"cache" "repertoire(0,set(2,1,0))"}
apply_effect "landscape" "rectangle(1,1,16,48) and (rand<5000)" { "cache" "repertoire(1,set(2,1))"}

To seed the center and right region with the desired distributions of agent classes and
identity repertoires:

apply_effect "landscape" "rectangle(17, 1, 32, 48) and (rand<10000)" {-1 "entrepreneur"}
apply_effect "landscape" "rectangle(17, 1, 32, 48) and (rand<2500)" {-1 "basic"}
apply_effect "landscape" "rectangle(17, 1, 32, 48) and (rand<2000)" {-1 "apathetic"}
apply_effect "landscape" "rectangle(17, 1, 32, 48) and (rand<10000)" {"cache" "repertoire(2,set(2,0))"}
apply_effect "landscape" "rectangle(17, 1, 32, 48) and (rand<5000)" {"cache" "repertoire(1,set(2,1,0))"}
apply_effect "landscape" "rectangle(33, 1, 48, 48) and (rand<10000)" {-1 "apathetic"}
apply_effect "landscape" "rectangle(33, 1, 48, 48) and (rand<2500)" { -1 "entrepreneur"}
apply_effect "landscape" "rectangle(33, 1, 48, 48) and (rand<2000)" {-1 "basic"}
apply_effect "landscape" "rectangle(33, 1, 48, 48) and (rand<10000)" { "cache" "repertoire(0,set(1,2,0))"}
apply_effect "landscape" "rectangle(33, 1, 48, 48) and (rand<5000)" { "cache" "repertoire(2,set(1,2))"}

To set the sight radius, rotation, substitution, substitution_activation triggers,
and influence level, of basic agents, entrepreneurs and apathetic:

set_routine_code_by_name "b_range" "1"
set_routine_code_by_name "b_level2" "2"
set_routine_code_by_name "b_level1" "5"
set_routine_code_by_name "b_level3" "7"
set_routine_code_by_name "b_influence" "1"
set_routine_code_by_name "e_range" "2"
set_routine_code_by_name "e_level2" "1"
set_routine_code_by_name "e_level1" "4"
set_routine_code_by_name "e_level3" "6"
set_routine_code_by_name "e_influence" "2"
set_routine_code_by_name "a_range" "3"

```
set_routine_code_by_name "a_level2" "4"
set_routine_code_by_name "a_level1" "8"
set_routine_code_by_name "a_level3" "12"
set_routine_code_by_name "a_influence" "0"
```

To save a snapshot with a desired name

```
save_state_to_file "ModelWith3AC.snp"
```

Modification of the simple script

We are now in a position to revisit the generic script we introduce in the begging of this manual, as well as the modifications discussed in previous sections. Notice, however, that unlike syntax reviewed earlier that that called and manipulated model4.mdl, the examples available in the next few pages refer to the snapshot *ModelWith3AC.snp*, a snapshot that one can generate using the syntax presented in the previous section. I urge to copy the syntax into a notepad, save it, open PS-I and load the file. You may also want to run each of the examples below to get better hands on experience with manipulating and running scripts.

Recaps and extensions

To run a simple loop using the 10 unique snapshots: load snapshot, run it 100 time
steps, and save statistics.

```
set x 100
for {set i 0} {$i < $x} {incr i 1} {
load_file "ModelWith3AC_$i.snp"
file delete "modelWith3AC_$i.csv"
set_filename "modelWith3AC_$i.csv"
for {set k 0} {$k < 100} {incr k 1} {step}
}
```

To create 10 unique versions of a snapshot: load snapshot, set random seeds, and save
snapshot

```
set x 100
for {set i 0} {$i < $x} {incr i 1} {
load_file "ModelWith3AC.snp"
set bias_seed [call_rand]
set_routine_code_by_name "bias_seed"  "$bias_seed"
set evolution_seed [call_rand]
set_routine_code_by_name "evolution_seed"  "$evolution_seed"
save_state_to_file  "ModelWith3AC_$i.snp"
}
```

To generate 10 versions of a snapshot designed with diagrams.[6]

```
set x 100
for {set i 0} {$i < $x} {incr i 1} {
load_file "diagram_generated_model.snp"
set landscape_seed [call_rand]
set_routine_code_by_name "landscape_seed"  "$landscape_seed"
reseed_model
save_state_to_file " diagram_generated_model _$i.snp"
}
```

To seed half of a rectangle array in the center of the model with apathetic agents with
an identity repertoire composed of identities 1 and 0, run it for 50 time steps, and loop
10 times.

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "ModelWith3AC_$i.snp"
apply_effect "landscape" "(rectangle(23, 23, 35, 35) and
(rand<5000)" {"cache  "repertoire(1, set(1,0))"  -1 "apathetic" }
file delete "ModelWith3AC(apathy)_$i.csv"
set_filename "ModelWith3AC(apathy)_$i.csv"
for {set k 0} {$k < 50} {incr k 1} {step}
}
```

To run a model for 40 time steps and then seed half of a rectangle array in the center
with apathetic agents with an identity repertoire composed of identities 1 and 0, run it
for 10 additional time steps, and loop 10 times.

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "ModelWith3AC_$i.snp "
file delete "ModelWith3AC(apathy)_$i.csv"
set_filename "ModelWith3AC(apathy)_$i.csv"
for {set k 0} {$k < 40} {incr k 1} {step}
apply_effect "landscape" "(rectangle(23, 23, 35, 35) and
(rand<5000)" {"cache  "repertoire(1, set(1,0))"  -1 "apathetic" }
for {set k 41} {$k < 50} {incr k 1} {step}
}
```

[6] All the necessary information for replication of models designed with the scripting tool diagrams (i.e. information about agent-classes, identity combinations and spatial configurations) is stored using a landscape seed. The script to generate version of a snapshot created with diagrams calls, and assigns a random landscape seed, and then reseeds the model to bring into effect a new version of the snapshot.

To set the maximum bias to 3, run the model for 50 time steps, and loop 10 times:

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "ModelWith3AC_$i.snp"
set_routine_code_by_name "bias_max" "3"
file delete "ModelWith3AC(apathy)_$i.csv"
set_filename "ModelWith3AC(apathy)_$i.csv"
for {set k 0} {$k < 50} {incr k 1} {step}
}
```

To set the maximum bias to 3 during the first 40 time steps, increase bias maximum to
5 during 10 additional time steps, and loop 10 times:

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "ModelWith3AC_$i.snp"
set_routine_code_by_name "bias_max" "3"
file delete "ModelWith3AC(apathy)_$i.csv"
set_filename "ModelWith3AC(apathy)_$i.csv"
for {set k 0} {$k < 40} {incr k 1} {step}
set_routine_code_by_name "bias_max" "5"
for {set k 41} {$k < 40} {incr k 1} {step}
}
```

The same as the previous example but combining information into one syntactic
argument.

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "ModelWith3AC_$i.snp"
set_routine_code_by_name "bias_max" "(time>40) ? 5 : 3"
file delete "ModelWith3AC(apathy)_$i.csv"
set_filename "ModelWith3AC(apathy)_$i.csv"
for {set k 0} {$k < 50} {incr k 1} {step}
}
```

To save statistical information for 10th and 50th time steps in two separate files:

```
set x 10
for {set i 1} {$i < $x} {incr i 1} {
load_file "ModelWith3AC.snp"
file delete "empty.csv"
```

```
set_filename "empty.csv"
for {set k 1} {$k < 10} {incr k 1} {step}
file delete "modelWith3AC(t=10)_$i.csv"
set_filename "modelWith3AC(t=10)_$i.csv"
for {set k 10} {$k < 11} {incr k 1} {step}
file delete "empty.csv"
set_filename "empty.csv"
for {set k 11} {$k < 50} {incr k 1} {step}
file delete "modelWith3AC(t=50)_$i.csv"
set_filename "modelWith3AC(t=50)_$i.csv"
for {set k 50} {$k < 51} {incr k 1} {step}
}
```

An experimental design using one script

Typically you may want to conceive and employ an experimental design to explore (or test hypotheses) on the effects of a variable or sets of variables (e.g. bias range and volatility combines) have on a specific outcome. In the following pages I shall offer two methods to organize an experimental protocol designed to study the impact on the evolution of the snapshot of increasing bias range and volatility during the first 10 time steps.

There are several ways of using a script to run an entire experimental design. In the conclusion I wish to introduce two common methods that you may to adopt and use to run elaborated experiment. Each of the two approaches give you significant control over the organization of the experimental design. You can change the text very easily by simple find and replace command.

Experimental design: Type one

In the first example PS-I is ordered to load a snapshot, generate a set of unique versions of the snapshot, run each of them and compile a set of baseline observations, and then generate observations for each level of the experimental treatment. You may find this type of script organization most useful if your goal is to conduct an experiment that involves a large number of experimental manipulations and changes. If that is your goal you may find that this type of script increases efficiency, improves the organization of experiment and the outcomes, and help minimizing syntactic errors.

ModelWith3AC_Full_Experiment.scp

To generate 10 unique versions of the snapshot:

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "ModelWith3AC.snp"
```

```
set bias_seed [call_rand]
set_routine_code_by_name "bias_seed"  "$bias_seed"
set evolution_seed [call_rand]
set_routine_code_by_name " evolution_seed"  "$evolution_seed"
save_state_to_file " modelWith3AC_$i.snp"
}
```

To run and compile a set of baseline observations:

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "ModelWith3AC_$i.snp"
set_routine_code_by_name "bias_max" "2"
set_routine_code_by_name "bias_min" "(-2)"
set_routine_code_by_name "bias_volatility" "500"
file delete "modelWith3AC(Bsln)_$i.csv"
set_filename "modelWith3AC(bsln)_$i.csv"
for {set k 0} {$k < 10} {incr k 1} {step}
set_routine_code_by_name "bias_max" "2"
set_routine_code_by_name "bias_min" "(-2)"
set_routine_code_by_name "bias_volatility" "500"
for {set k 0} {$k < 90} {incr k 1} {step}
save_state_to_file " modelWith3AC(bsln)(t100)_$i.snp"
}
```

To run and collect data on the first experimental condition (bias 3, (-3), Volatility
1000):

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "ModelWith3AC_$i.snp"
set_routine_code_by_name "bias_max" "3"
set_routine_code_by_name "bias_min" "(-3)"
set_routine_code_by_name "bias_volatility" "1000"
file delete "modelWith3AC(B3V1000)_$i.csv"
set_filename "modelWith3AC( B3V1000)_$i.csv"
for {set k 0} {$k < 10} {incr k 1} {step}
set_routine_code_by_name "bias_max" "2"
set_routine_code_by_name "bias_min" "(-2)"
set_routine_code_by_name "bias_volatility" "500"
for {set k 0} {$k < 90} {incr k 1} {step}
save_state_to_file " modelWith3AC(B3V1000)(t100)_$i.snp"
}
```

To run and collect data on the first experimental condition (bias 4, (-4), Volatility
2000):

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
load_file "ModelWith3AC_$i.snp"
set_routine_code_by_name "bias_max" "4"
set_routine_code_by_name "bias_min" "(-4)"
set_routine_code_by_name "bias_volatility" "2000"
file delete "modelWith3AC(B4V2000)_$i.csv"
set_filename "modelWith3AC( B4V2000)_$i.csv"
for {set k 0} {$k < 10} {incr k 1} {step}
set_routine_code_by_name "bias_max" "2"
set_routine_code_by_name "bias_min" "(-2)"
set_routine_code_by_name "bias_volatility" "500"
for {set k 0} {$k < 90} {incr k 1} {step}
save_state_to_file " modelWith3AC(B4V2000)(t100)_$i.snp"
}
```

Experimental design: Type two

The second type of scripts consolidates all the experimental conditions within one long loop. The script orders PS-I to load a snapshot generate a unique version, and run through all the experimental conditions. Then PS-I loads the snapshot again, generates a second unique version and runs through all the conditions. You may find this type of scripts more useful for a proof of concept and a "quick and dirty" test of the effects of the variable(s) you wish to study. Using this method you may conduct a swift run through the big sections of a parameter space, in order to test the robustness of some observations. Surely, you can use this type of scripts for running an entire experiment, but you may find that writing a script of this type requires somewhat more experience and may be more conducive to syntactic errors.

<u># ModelWith3AC_All_in_one_loop.scp</u>

To set the number of repetitions and open up a loop:

```
set x 10
for {set i 0} {$i < $x} {incr i 1} {
```

To generate and save the first unique version of a pre-designed snapshot:

```
load_file "ModelWith3AC.snp"
set bias_seed [call_rand]
set_routine_code_by_name "bias_seed"  "$bias_seed"
set evolution_seed [call_rand]
```

```
set_routine_code_by_name " evolution_seed"  "$evolution_seed"
save_state_to_file " modelWith3AC_$i.snp"
```

To load the first unique version and set baseline conditions:

```
load_file "ModelWith3AC_$i.snp"
set_routine_code_by_name "bias_max" "2"
set_routine_code_by_name "bias_min" "(-2)"
set_routine_code_by_name "bias_volatility" "500"
file delete "modelWith3AC(Bsln)_$i.csv"
set_filename "modelWith3AC(bsln)_$i.csv"
for {set k 0} {$k < 10} {incr k 1} {step}
set_routine_code_by_name "bias_max" "2"
set_routine_code_by_name "bias_min" "(-2)"
set_routine_code_by_name "bias_volatility" "500"
for {set k 0} {$k < 90} {incr k 1} {step}
save_state_to_file " modelWith3AC(bsln)(t100)_$i.snp"
```

To load the first unique version and set the first experimental conditions:

```
load_file "ModelWith3AC_$i.snp"
set_routine_code_by_name "bias_max" "3"
set_routine_code_by_name "bias_min" "(-3)"
set_routine_code_by_name "bias_volatility" "1000"
file delete "modelWith3AC(B3V1000)_$i.csv"
set_filename "modelWith3AC(B3V1000)_$i.csv"
for {set k 0} {$k < 10} {incr k 1} {step}
set_routine_code_by_name "bias_max" "2"
set_routine_code_by_name "bias_min" "(-2)"
set_routine_code_by_name "bias_volatility" "500"
for {set k 0} {$k < 90} {incr k 1} {step}
save_state_to_file " modelWith3AC(B3V1000)(t100)_$i.snp"
```

To load the first unique version and set the second experimental conditions:

```
load_file "ModelWith3AC_$i.snp"
set_routine_code_by_name "bias_max" "4"
set_routine_code_by_name "bias_min" "(-4)"
set_routine_code_by_name "bias_volatility" "2000"
file delete "modelWith3AC(B4V2000)_$i.csv"
set_filename "modelWith3AC(B4V2000)_$i.csv"
for {set k 0} {$k < 10} {incr k 1} {step}
set_routine_code_by_name "bias_max" "2"
set_routine_code_by_name "bias_min" "(-2)"
set_routine_code_by_name "bias_volatility" "500"
for {set k 0} {$k < 90} {incr k 1} {step}
```

save_state_to_file " modelWith3AC(B4V2000)(t100)_$i.snp"

To end the loop and return to beginning:

}

A final Note

In the introduction to this manual I introduced scripts as an important supplementary method that may help you, the user, to design and conduct extensive and elaborated experiments with PS-I. Scripts can be used to design snapshots and/or produce any number of repetitions of a pre-designed snapshot or model without requiring you to guide the process step by step. Scripts allow researchers to conceive of and employ extensive experimental designs without stopping and restarting the experiment in order to set different parameter values to independent and interaction effects. In other words, a script is a powerful tool with extensive (and some yet unexplored) capabilities.

It is important to note that this manual does not include a comprehensive list of all the possible syntactic combinations available using scripts. Rather it was my intention here to give you just an idea on the range of possibilities, and to urge you to change/adapt the syntax to your particular experimental needs.